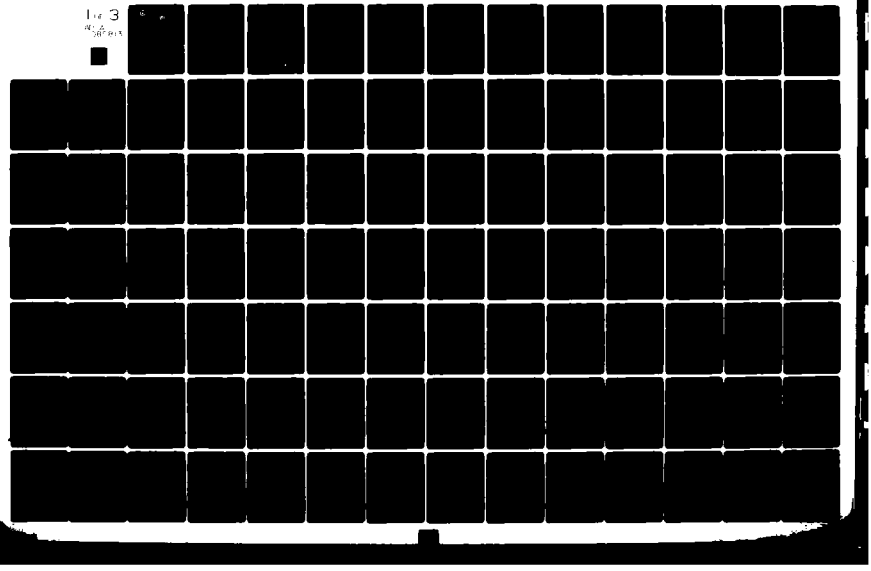
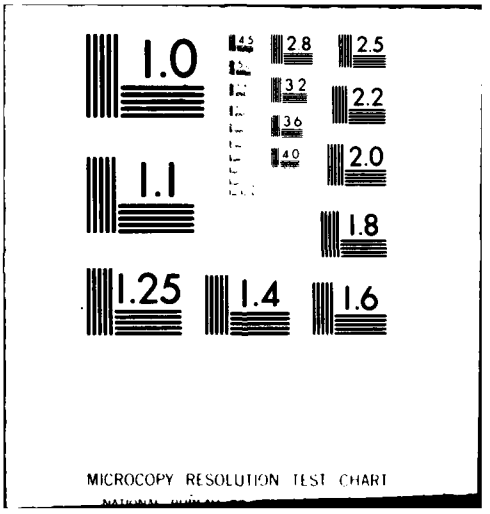
AD-A085 813 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON DC F/G 15/7
THE CCTC QUICK - REACTING GENERAL WAR GAMING SYSTEM (QUICK) PRO--ETC(U)
MAY 80
UNCLASSIFIED CCTC-CSM-MM-9-77-V1-CHG-3 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART

DEFENSE COMMUNICATIONS AGENCY
NATIONAL MILITARY COMMAND SYSTEM
SUPPORT CENTER
WASHINGTON, D. C. 20301

LEVEL III

IN REPLY
REFER TO: C314

22 MAY 1980

TO: RECIPIENTS

SUBJECT: Change 3 Program Maintenance Manual CSM MM 9-77, Volume I, Data Management Subsystem

1. Insert the enclosed change pages and destroy the replaced pages according to applicable security regulations.

2. Also enclosed are pages 160, 559, and 676.15 through 676.18 change 1. They were originally printed incorrectly.

3. A list of Effective Pages to verify the accuracy of this manual is enclosed. This list should be inserted before the title page.

4. When this change has been posted, make an entry in the Record of Changes.

FOR THE DIRECTOR:

252 Enclosures
Change 3 Pages

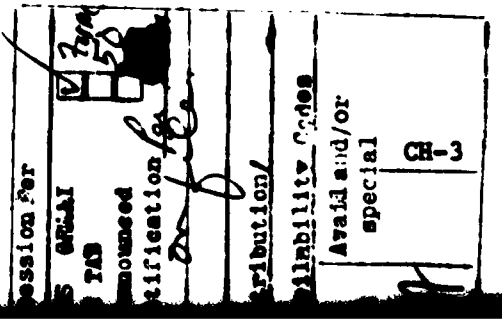
J. DOUGLAS POTTER
Assistant to the Director
for Administration

EFFECTIVE PAGES - APRIL 1980

This list is used to verify the accuracy of CSM MM 9-77, Volume I after change 3 pages have been inserted. Original pages are indicated by the letter O, change 1 pages by the numeral 1, change 2 pages by the numeral 2, and change 3 pages by the numeral 3.

Page No.	Change No.	Page No.	Change No.
Title Page, Part I	0	62	2
ii	3	63-75	0
iii-iv	1	76	3
v	3	77-79	1
vi-vii	1	79.1-79.3	3
viii	2	79.4	1
ix-x	0	80-82	0
xi	1	83	3
1-2	0	84-88	0
3	3	89-90	3
4	1	90.1-90.2	1
5-14	0	91	2
15	3	92	0
16	1	93	3
17	0	94-99	0
18	1	100-101	1
19-22	0	102	3
23	3	103-126	1
24-29	0	127	3
30	1	128-130	1
31	2	131	3
32-33	0	132-133	0
34	2	134-135	0
35	0	136-138	2
36-38	3	139-144	0
39-40	2	145	3
41	1	146	0
41.1-41.2	1	147	3
42-43	3	148-150	1
44-45	0	151-152	0
46	1	153-154	2
47	2	155	1
48-49	3	156	3
50	1	157-158	0
51	2	159-165	1
52-55	3	166	3
56	2	167-168	0
57-60	0	169-172	1
61	3		

Page No.	Change No.	Page No.	Change No.
173	0	308	3
174	1	309	0
175	0	310	2
176	1	311-316	0
177-181	0	317-318	3
182	2	319-321	0
183	3	322	3
184-185	0	323-327	0
186	3	328	3
187-188	0	329-332	0
189	3	333	3
190-218	0	334-335	0
219	3	336-342.2	3
220	0	343-365	0
221	3	366-367	3
222-242	0	368	0
243	3	369	3
244-246	0	370-381	0
247	3	382	3
248	0	383-434	0
249-250	3	434.1	3
251	0	434.2	1
252	3	434.3	3
253-254	0	434.4	3
255	2	Title Page, Part II	0
256-257	0	ii	3
258-260	3	iii	1
261-264	0	iv-ix	3
265-265.2	3	x	0
266	0	xi	1
267	2	435-438	0
268	0	439	1
269	3	440-444	0
270-275	0	445	1
275.1-275.8	3	446-490	0
276-284	0	491	3
285	3	492-512	0
286-290	0	513	3
291	3	514-546	0
292-294	0	547-548	1
295-296	3	549-558	0
297-301	0	559	1
302	3	560-572	0
303-304	0	572.1-572.6	1
305	3		
306-307	0		



Page No.	Change No.	Page No.	Change No.
573-580	0	764	2
581	1	765-766	0
582-589	2	767-768	2
590	0	769	3
591-593	1	770-780	0
593.1-593.2	1	780.1-780.2	2
594-595	1	781-782	0
596	0	783-785	2
597-598	3	786-791	0
599-605	0	792-793	2
606	3	794-800	0
607-632	0	801-802	2
633	2	803-804	0
634-661	0	804.1-804.2	3
662	3	805-810	0
663-665	0	811-815	2
666	2	816-829	0
667	0	830	3
668	2	831-839	0
669-675	0	840-841	2
676	2	842-847	0
676.1-676.20	1	848	3
676.21	2	849	3
676.22-676.38	1	852	3
677-678	0	853-855	0
679	2	856-857	2
680	0	857.1-857.6	2
681	3	858-868	0
682	2	869	3
683-694	0	870-871	0
695-698	2	872-873.4	3
698.1-698.2	3	874-890	0
699	2	891	3
700-712	0	892	2
713-714	2	893-905.4	3
715-718	0	906-910	3
719	1	910.1-910.2	3
720	0	911	3
721	1	911.1-911.2	3
722-742	0	912-919.8	3
732-744	1	920-921.8	3
745-752	0	922	0
753	2	923	3
754-755	0	924	0
756	3	925-926	3
757-763	0		

ACKNOWLEDGMENT

This documentation was prepared under the direction of the Chief for Military Studies and Analysis, CCTC, in response to a requirement of the Studies, Analysis, and Gaming Agency, Organization of the Joint Chiefs of Staff. Technical support was provided by System Sciences, Incorporated under Contract Number DCA 100-75-C-0019. Change set two was prepared under Contract Number DCA 100-78-C-0035. Computer Sciences Corporation prepared change set three under Contract Number DCA 100-78-C-0042.

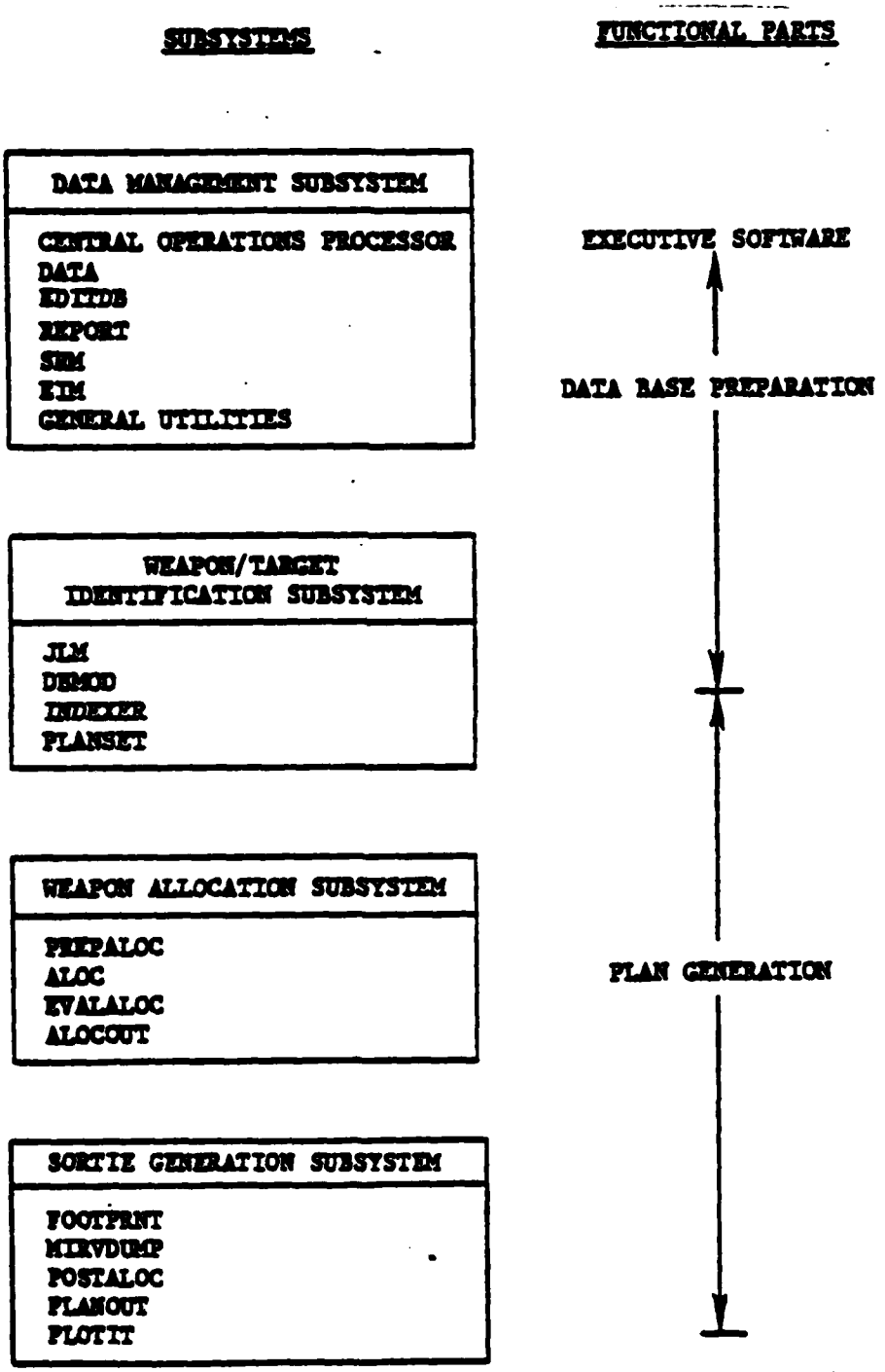


Figure 1. Major Subsystems of the QUICK System

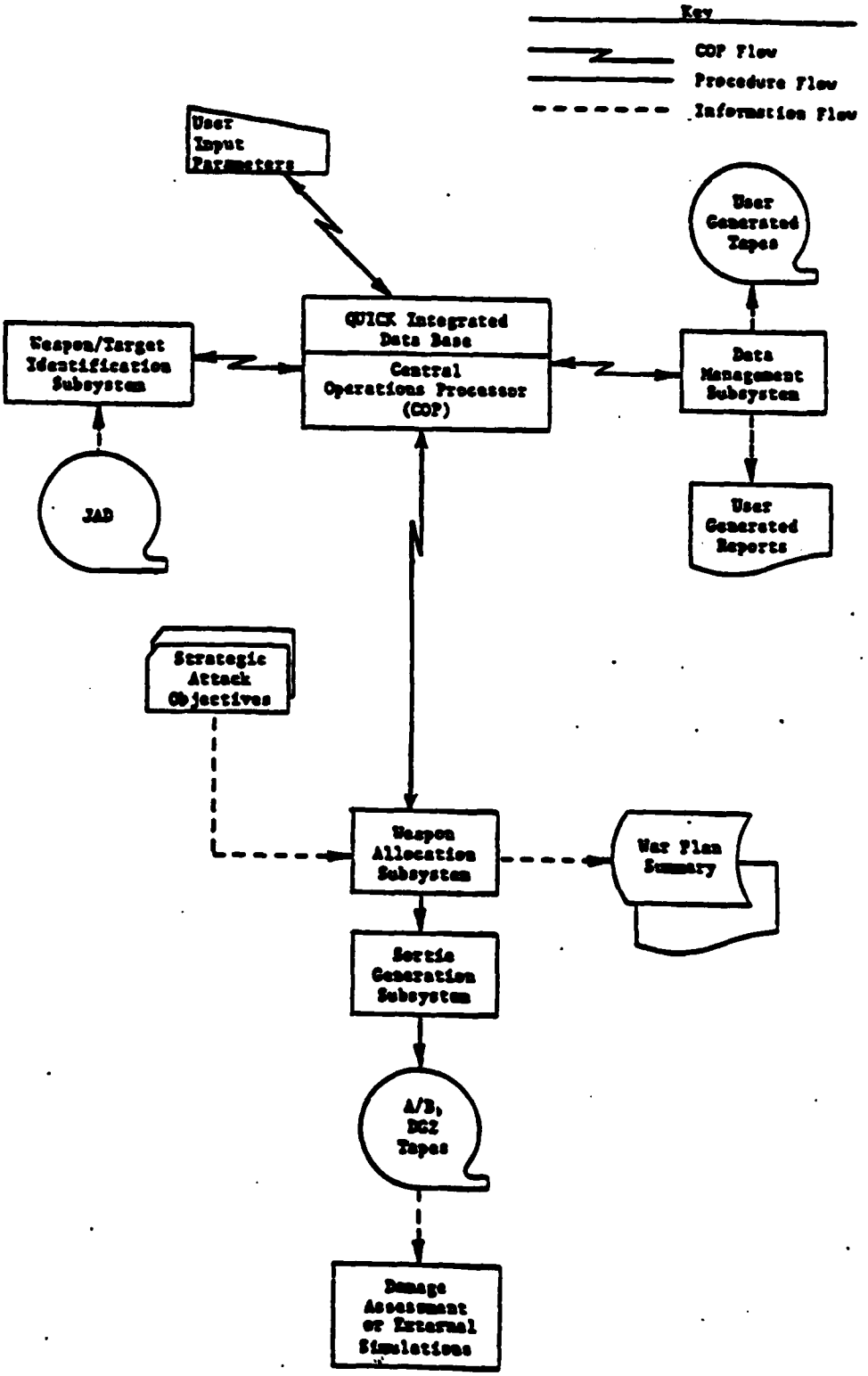


Figure 2. Procedure and Information Flow in QUICK/HIS 6000

Table 1. COP Entry Points (Part 1 of 2)

ENTRY POINT NAME	ARGUMENTS	DESCRIPTION
CLEANP	(none)	IDS Buffer Flush (.QFLSH)
CLZIDS	(none)	Close IDS File
DIRECT	(none)	Retrieves array IDS record based on its binary reference code stored in common C10
DLETE	Record Type Name	Deletes current record of type named
ERPRIN	(none)	Process input line for error message
HDFND	BCD reference Code, CLASS value, SIDE value, Record Type Name	Finds BCD reference code, given values for CLASS, SIDE and/or Record Type Name
HDPUT	BCD reference Code, CLASS value, SIDE value, Record Type Name	Adds New Header of Type named with given values for CLASS and SIDE
HEAD	Chain Name	Retrieves master of chain named
INPRIN	(none)	Process input line normally
INSDEL	(none)	Deletes all input tables
INSFLS	(none)	Assures that any additions to input tables are recorded
INSGET	Array to contain output, Index of first item to retrieve, Number of items to retrieve	Obtains input instructions from input tables
INSPUT	Array which contains items to be inserted, Index to input table (should be set to start point minus one, will be returned as end point), Number of items to add	Inserts items in input tables
LGPRIN	(none)	Process input line from long string

Table 1. (Part 2 of 2)

ENTRY POINT NAME	ARGUMENTS	DESCRIPTION
MODFY	Record Type Name	Modifies current record of type named to reflect current values in common
NEXTTT	Chain Name	Retrieves next record of chain named
OPNIDS	(none)	Opens IDS file
RETRV	Record Type Name	Retrieve record of type named (should be used only for primary or CALC records)
STORE	Record Type Name	Store new record of type named

Table 3. Instruction Code Bit Configuration

BITS	DESCRIPTION
29-32	General instruction code 0 = Terminator and Operator follows 1 = General Item Follows 2 = Equals 3 = Greater Than, or Greater Than or Equal To 4 = Less Than, or Less Than or Equal To 5 = Logical Operations 6 = Loads or Stores 7 = Adds or Subtracts 8 = Multiplies or Divides 9 = Powers
33-35	For general instruction code = 0, 1, or 5: Particular Code Discriminator
33	For general instruction code = 3, 4, 6, 7, or 8: Second Code Discriminator 0 = First operation shown above 1 = Second operation shown above
34-35	Value Type Discriminator 1 = Alphabetic value 2 = Numeric value 3 = Internal variable

Table 4. Input Instruction Formats (Part 1 of 3)

ARRAY NUMBER	DESCRIPTION
	End of Clause
I	='1', instruction code
	End of Phrase
I	='2', instruction code
	Operator Follows
I	='3', instruction code
I+i	Operator number
	LIKE String Follows - Alphabetic Identifier
I	='11', instruction code
I+1	Identifier attribute address
I+2	Identifier attribute number
I+3	='9', instruction code
I+4	='9', instruction code
I+5	First half of identifying value
I+6	Second half of identifying value
	LIKE String Follows - Numeric Identifier
I	='11', instruction code
I+1	Identifies attribute address
I+2	Identifies attribute number
I+3	= '10', instruction code
I+4	= '10', instruction code
I+5	Floating point identifying value
	Long String Follows
I	='13', instruction code
I+1	Number of characters in long string
I+2	First six characters of first pair
I+3	Second six characters of first pair
.	
.	
.	
I+(N-1)	First six characters of last pair
I+N	Second six characters of last pair

Table 8. (Part 5 of 5)

ARRAY NUMBER	ARRAY VALUE	DESCRIPTION
86	0	
87	50*	Load Numeric
88	6	Numeric is an attribute
89	157	Attribute's address (SPDLO)
90	141	Attribute's number (SPDLO)
91	0	No OF phrase
92	18*	Set equal to numeric
93	10	Numeric is a constant
94	900.	Numeric constant
95	2*	End of phrase
96	1*	End of clause

*Instruction code.

- Record Name - COBOL name for the record
- Number of Records - Estimated number of data records denoted by the record block
- Record Length - Number of characters requested for the block
- Double Line - Denotes CALC records

Individual record blocks are connected with vectors and arrows representing each chain relationship in the file. Beside each chain vector, the chain name is entered. Below each chain name an indication is given to describe how records are positioned or entered in the chain. The appropriate entries are: 'F' for First, 'L' for Last, 'S' for Sorted, 'B' for Before, and 'A' for After.

Organizationally, the QUICK integrated data base may be divided into two parts: the scenario (or gaming) data and the organization data.

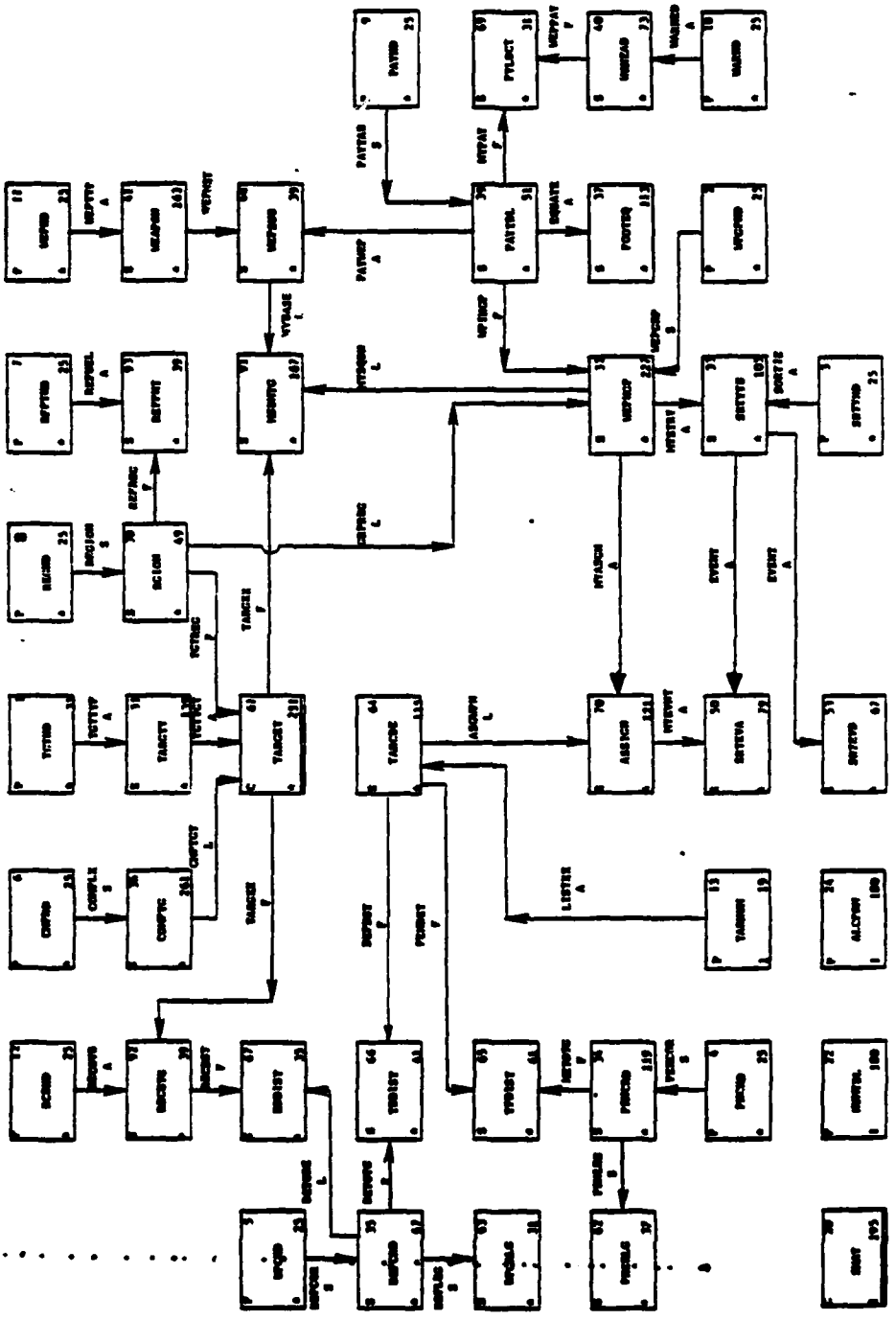
2.4.1 Scenario Data Structure. Figure 6 is a picture of the scenario data structrue. However, as this structure is quite complex, it will be divided into four parts for discussion purposes. The figures given for the subdivisions are incomplete in that they do not have connected to them the chains which interrelate the four subdivisions.

2.4.1.1 Target Data Structure. Figure 7 shows the target data organization. The TARGET record is the central record type of the data base and is a CALC record. The principal hierarchy is that of target class header (TGTHD) target type (TARGTY) and individual target (TARGET). Targets are grouped by region and complex. The TARGXX chain links some individual target records to additional data. In one case the data is that for a recovery base (RECBTG). In the other case the target is also a missile, bomber, or tanker base (MSBMTG).

Figure 7 also shows refuel points (REFPNT) associated with their region.

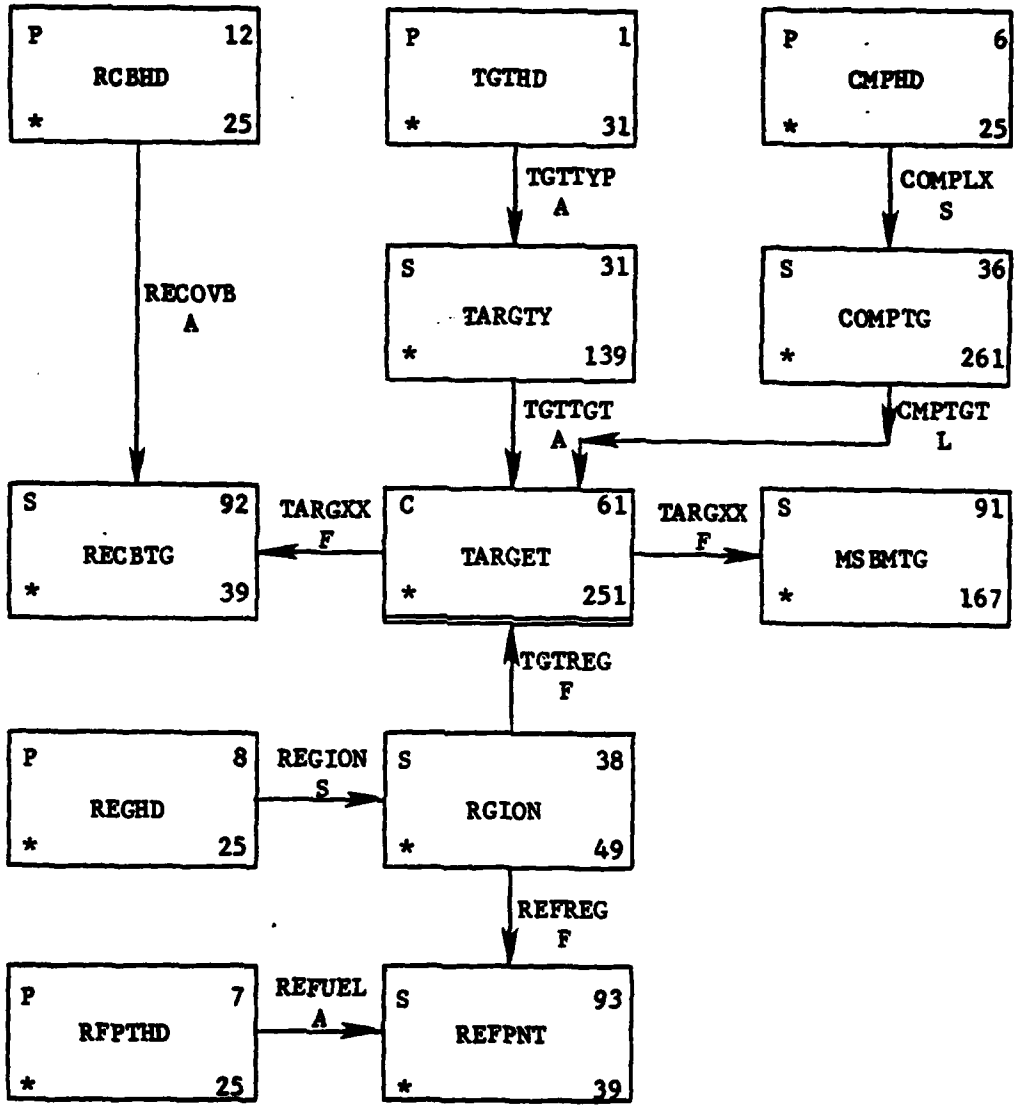
2.4.1.2 Weapon Data Structure. Figure 8 shows the weapon data structure. One hierarchy contains the weapon class (WEPHD), weapon type (WEAPON), weapon type subdivided by payload (WEPSUB) and the individual weapon base (MSBMTG). There is also a warhead class (WARHD) with warhead type as details (WRHEAD). Weapons are connected to their warheads via a payload table (PAYTBL) which is master of one chain containing all weapon subtypes (WEPSUB) which utilize that table and another chain which contains counts (PYLDCT) of the various warhead types.

Finally, the QUICK system creates weapon groups (WEPNGP) which have a payload table and a number of individual bases (MSBMTG). These groups are also assigned a geographic region.



*Number of records is scenario dependent.

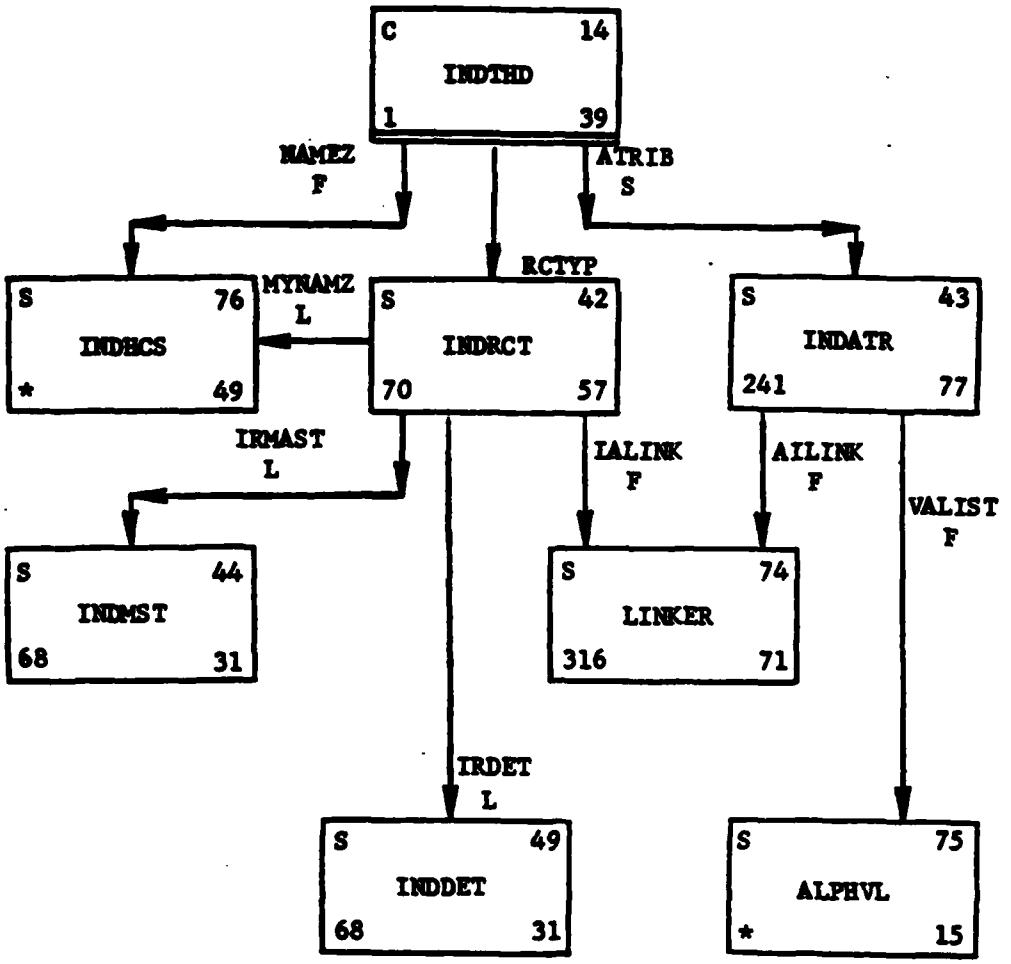
Figure 6. Scenario Data Structure



*Number of records is scenario dependent

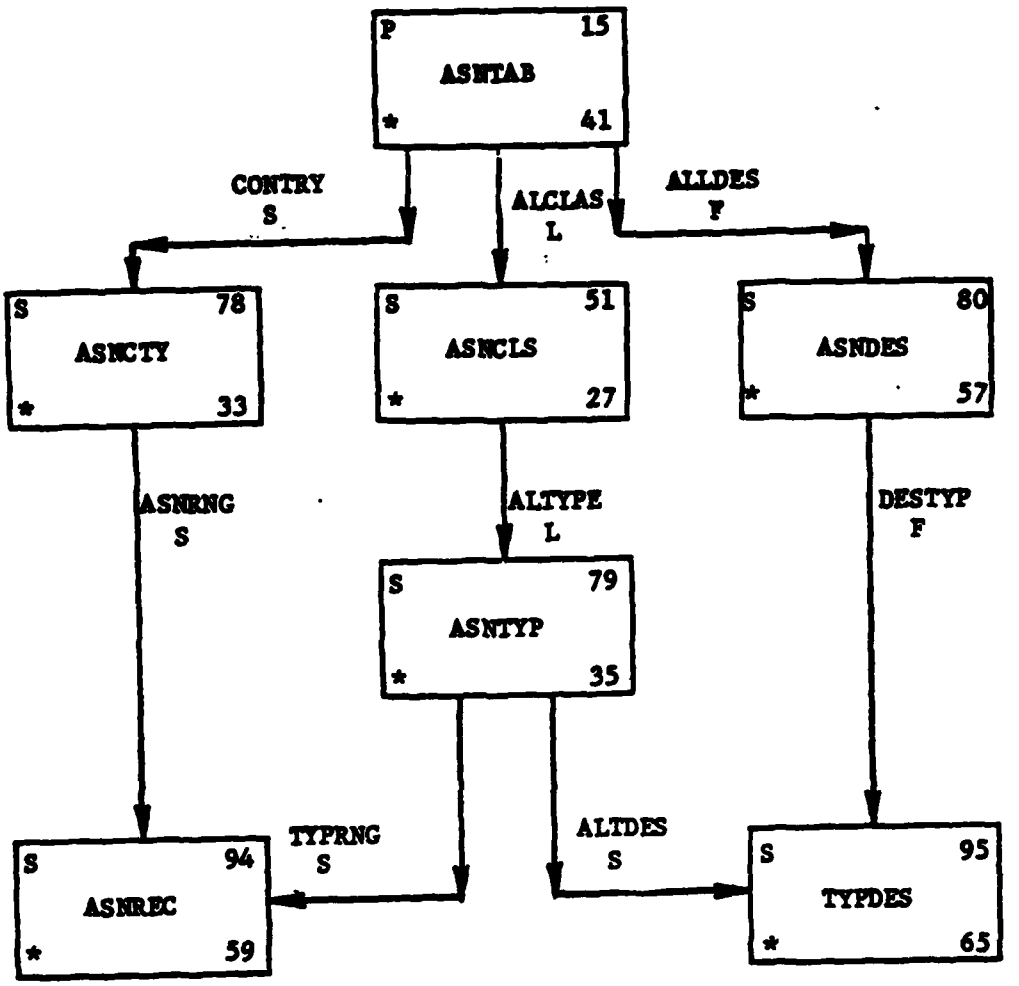
Figure 7. Target Data Structure

country codes and the region they are in (ASNCTY). These records are sorted on region and country code. Also under each header, are records (ASNCLS) containing the values for CLASS for the side. Under each of these records are all the TYPE names that belong to this class (ASNTYP). The countries and types are connected via common ASNREC records. These records contain restrictions based on Category Code, the location or owner of the target, and either its name or size. It also contains the TASK that will be assigned to a target meeting these restrictions. On the other chain under TYPE are the alphabetic portions of DESIG to be used in assigning a DESIG to the target (TYPDES). Subsequent DESIGs are used if the first values are already used. All of these records with the same alphabetic portion of DESIG are chained together under a common record (ASNDES) containing the DESIG and the number in each region.



* Number of Records is scenario dependent.

Figure 10. Data Organization Index



* Number of records is scenario dependent.

Figure 11. Assignment Table

2.4.2.3 Miscellaneous Organizational Data. Figure 12 shows a number of smaller data organizations. The largest of these is the syntax directory. One header (SYNHD) has a record chained to it for each verb (SYNVBB). The other header (ADVHD) has a record chained to it for each adverb (PRMADV). The adverb record contains data which describes the clause and phrase type. If the phrase type is 'element' the adverb has chained to it records describing the legal elements (ADVELM). Finally a record type links verbs to their legal adverbs (SYNCLZ).

The module link table (MODTAB) is a single record which is on no chains. The dictionary is a hierarchy with the header (DCTHD), tab characters (DCTTAB) and words with that tab character (DCTWRD).

The display table hierarchy is the header (DISPHD), the individual display name (DISPRC) and the elements which make up the display (DISPDT).

Finally, the utility table (TABLEZ) are chained to their header (TABLST). Actually, utility tables are primary records which the using routines create and maintain as additional storage areas. The header exists to assure that all utility tables may be deleted once they are no longer needed.

2.4.3 Data Base Record Content. Previous subsections defined QUICK's IDS structure. This subsection gives the definition of what words or attributes are contained within each data record.

2.4.3.1 Primary Records. Primary records, called headers, are used as data base entry points. These headers are identified through a value for the attribute CLASS plus, in most cases, a value for the attribute SIDE. A list of all QUICK data base headers appear in table 9 along with the header's record type number, what attributes need to be set and a list of allowable CLASS values for each header.

2.4.3.2 Secondary Records. The remainder of the record types are secondary records (the TARGET record is a CALC record). Each secondary record is a collection of attributes and/or internally used values (table 10).

A list of QUICK data base chains appears in table 11. All chains in the data base structure are 'linked to prior' so that when any record type is deleted, the physical file space it occupies is released for other use. Many of the chains are also 'linked to master' to speed processing when a subroutine calls entry point HEAD for those chains (see table 12).

Table 9. (Part 2 of 2)

RECORD TYPE NUMBER	RECORD TYPE NAME	DESCRIPTION (ATTRIBUTES)	CLASS VALUES
14	INDTHD	Data Organization Index Header (CLASS) (This is a CALC record)	INDEX
15	ASNTAB	Assignment Table Header (CLASS,SIDE)	ASSIGN
16	DCTHD	Dictionary Header (CLASS)	DICTON
17	SYNHD	Syntax Directory Verb Header (CLASS)	SYNTAX
18	ADVHD	Syntax Directory Adverb Header (CLASS)	ADVERB
19	MODTAB	Module Link Table (CLASS, link Table (100 words))	MODTAB
20	SMAT	SMAT Array (CLASS) (This is a CALC record)	SMAT
21	DISPHD	REPORT Module Display Header (CLASS)	DISPLY
22	NUMTBL	General Number Table (CLASS)	NUMBER
23	TABLST	Utility Table Header (CLASS) (This is a CALC record)	TABLST
24	ALCPRM	ALOC Control Parameters (CLASS)	ALCPRM

Table 10. Data Base Record Types - Secondary Records
(Part 1 of 4)

RECORD NUMBER	TYPE NAME	DESCRIPTION (ATTRIBUTES)
31	TARGTY	Target Type (CNTRYLOC, CNTRYOWN, FLAG, FVALT1, FVALT2, FVALT3, FVALT4, FVALT5, FVULN1, NHRDCOMP, NTIMCOMP, T1, T2, T3, T4, T5, TYPE, VULN1 VULN2)
32	WEPNGP	Weapon Group (ATTINC, EXPASM, GBASE, GFRASM, GLAT, GLONG, GNWPNADJ, GNWPNS, GNVEH, GPKNAV, GREFCODE, GREFTIME, GROUP, GSBL, GSBLREAL, GSTART, GTYPE, GTYPREFC, GYIELD, IALERT, IREG, MAXSAL, NFIXES, NSAL, NSFIX, NUMALOC) (GROUP is used as a Match Key)
33	SRTYTB	Sortie Table (SDELAY, SDEPEN, SINDEXNO, SLAT, SLONG, SLOW, SLOW1, SLOW2, SLOW3, SORTNO, SREFUEL, SVEHNUM)
34	PENCRD	Penetration Corridor (ATTRCO, ATTRPRE1, ATTRPRE2, ATTRPRE3, ATTRSU, CORNUM, DEFDIST1, DEFDIST2, DEFDIST3, DEFRAN, HILOAT, KORSTY, NPRCRDEF, ORLAT, ORLONG)
35	DEPCRD	Depenetration Corridor (CORNUM, MYRECOV1, MYRECOV2, MYRECOV3, MYRECOV4)
36	COMPTG	Complex Target (CNTRYLOC, CNTRYOWN, DESIG, FLAG, FVALT1, FVALT2, FVALT3, FVALT4, FVALT5, FVULN1, HAZ, HAZ2, HGZ, HGZ2, ICOMPL, IDHOB, INDEXNO, LAT, LONG, MAXFRA, MAXKILL, MINKILL, MISDEF, NAME, NHRDCOMP, NTIMCOMP, NTINT, RADIUS, TARDEFHI, TARDEFLO, TASK, TGTMULT, TGTNUMB, T1, T2, T3, T4, T5, VALUE, VOZ) (ICOMPL is a match-key)
37	FOOTEQ	Footprint Equation (Contains one hundred words which are module defined)
38	RGION	Region (IREG, CCREL) (IREG is a match key)
39	PAYTBL	Payload table (PAYTBLNM)
40	WRHEAD	Warhead (CEPASM, CPASMZRO, FFRAC, NAREADEC, NCMS, NDECOYS, NWHDS, PAYALT, PDUD, RANGEASM, RELASM, SPEEDASM, TYPE, YIELD)

Table 10. (Part 2 of 4)

RECORD NUMBER	TYPE NAME	DESCRIPTION (ATTRIBUTES)
41	WEAPON	Weapon type (ACTIVE, ALTDLY, BALC, CEP, CEPMIN, CMISS, FUNCTI, IPENMO, IRECMO, IREP, LCHINT, MAXSAL, NALTDLY, NMPSIT, PDES, PFPF, PINC, PLABT, RRABT, RANGE, RANGEDEC, RANGEREF, REANG, REL, RNGMIN, SIMLUN, SLOPE, SPDLO, SPEED, TOFMIN, TTOS, TYPE)
42	INDRCT	Index Record Type Record (contains record type name and number)
43	INDATR	Index Attribute Record (ATTRIBN1, ATTRIBN2, ATTRIBNO, ATTRBTYP, ATTRIBAD, ATDEFALT, ATTRNGHI, ATTRNGLO) (ATTRIBN1 and ATTRIBN2 are match keys)
44	INDMST	Index Master Record (CHAINNAM, MASDETNM, MASDETNO)
45	DCTTAB	Dictionary Tab-character (TABCHAR) (TABCHAR is a match key)
46	SYNVRB	Syntax Verb (CLAUSESW, VERBVAL) (VERBVAL is a matcky key)
47	PRMADV	Syntax Adverb (ADVERBVL, CLAUSETY, PHRASETY) (ADVERBVL is a match key)
48	TABLEZ	Utility Table (contains 100 words which are module defined)
49	INDDET	Index Detail Record (CHAINNAM, MASDETNM, MASDETNO)
50	SRTEVA	Sortie Event Type A (LAT, LONG, SCUMSURV, SCHANGE, SDAMEXP, SDELTIME, SEVCODE, SLOCATTR, SPLACE)
51	ASNCLS	Assignment Table Class (ACLASS)
52	DISPRC	Display Record (DISPNAM1, DISPNAM2)
53	SRTEVB	Sortie Event Type B (Same attributes as record number 50)
61	TARGET	Target (BENO, CATCODE, DESIG, HAZ, HAZ2, HGZ, HGZ2, ICOMPL, IDHOB, IGIW, INDEXNO, IREG, ISITE, LAT, LONG, MAJOR, MAXFRA, MAXKILL, MINKILL, MINOR, MISDEF, NAME, NTINT, POP, RADIUS, SIDE, TARDEFHI, TARDEFLO, TASK, TGTNUMB, VALUE, VOZ, WACNO) (Record is a CALC record - randomized on DESIG)

Table 10. (Part 3 of 4)

RECORD NUMBER	TYPE NAME	DESCRIPTION (ATTRIBUTES)
62	PNCRLG	Penetration Corridor Leg (ATTRLE, DOGLEG, LAT, LONG)
63	DPCRLG	Depenetration Corridor Leg (DOGLEG, LAT, LONG)
64	TARCDE	Target List Element (TGTNUMB, TGTREFCD)
65	TPDIST	Distance from Target to Penetration Corridor (ATTRCD, DISTANCE)
66	TDDIST	Distance from Target to Depenetration Corridor (DISTANCE, DISTDF)
67	RDDIST	Distance from Recovery Base to Depenetration Corridor (DISTANCE)
68	WEPSUB	Weapon Subtype (PAYTBLNM) (PAYTBLNM is a match key)
69	PYLDCT	Count of warhead type in Payload Table (NUMLOAD)
70	ASSIGN	Assignment of Weapon Group to Target (ARRIVE, ASGHOB, DGZLAT, DGZLONG, FIXED, FLMULT, FSALVO, GROUP, KORR, OFFLAT, OFFLONG, PEN, RVAL, TGTNUMB)
71	DCTWRD	Dictionary Word (WORDSTR1, WORDSTR2, WORDTY, WORDVL)
72	SYNCLZ	Links Verbs to Adverbs (ADVERBVL, VERBVAL)
73	ADVELM	Gives Legal Elements for elemental Adverbs (ELEMNTTY, ELEMNTVL)
74	LINKER	Links Attributes to Record Types (ATTRIBN1, ATTRIBN2, ATTRIBAD, ATTRBTYP, contains 2 words which are not attributes)
75	ALPHVL	Legal Values for 'LIST' Attributes (ALPLSTVL)
76	INDHCS	Stores Header reference codes (contains header reference code (two words), and the associated CLASS and SIZE values)
77	DISPDT	Display table list (contains 100 words which are created by REPORT)

Table 10. (Part 4 of 4)

RECORD NUMBER	TYPE NAME	DESCRIPTION (ATTRIBUTES)
78	ASNCTY	Assignment Table Country (COUNTRY, REGION)
79	ASNTYP	Assignment Table Type (ATYPE)
80	ASNDES	Assignment Table Desig (DESIGA2, KOUNT1, KOUNT2, KOUNT3, KOUNT4, KOUNT5)
91	MSBMTG	Missile Bomber Target (ADBLI, ADBLR, ALRTDB, ALRTDL, GROUP, IREFUEL, NADBLI, NADBLR, NLRTDB, NLRTDL, NOALER, NOINCO, NOPERSQ, NPRSQ1, NPRSQ2, NPRSQ3, NPRSQ4, NUMDBL, PAYTBLNM, PKNAV, VONBASE, WEPNAME)
92	RECBTG	Recovery base (CAPACITY)
93	REFPNT	Refuel Point (LAT, LONG, IREG)
94	ASNREC	Assignment Table Category (ASNTASK, CATHI, CATLO, CNFLG, MINCAP
95	TYPDES	Assignment Table Type DESIG (DESIGA2, FULL1, FULL2, FULL3, FULL4, FULL5)

Table 11. Data Base Chains (Part 1 of 4)

CHAIN NAME	MASTER RECORD	DETAIL RECORD	DESCRIPTION
ADVADV	ADVHD	PRMADV	Links Adverb Header to adverbs
ADVERB*	PRMADV	SYNCLZ	Links adverb to link to verb
AILINK*	INDATR	LINKER	Links Attribute Record to link to record type
ALCLAS	ASNTAB	ASNCLS	Links Assignment table header to assigned classes
ALLDES	ASNTAB	ASNDES	Links Assignment table header to assigned DESIG
ALTDES*	ASNTYP	TYPDES	Links Assigned type to link to DESIGs for that type
ALTYPE	ASNCLS	ASNTYP	Links Assigned class to assigned types in that class
ASGWPN*	TARCDE	ASSIGN	Links target to fix assignments to it
ASNRNG*	ASNCTY	ASNREC	Links Assigned country to category ranges for that country
ATRIB	INDTHD	INDATR	Links index header to attribute record
CLAUSE	SYNVRB	SYNCLZ	Links verb to link to adverbs
CMPTGT*	COMPTG	TARGET	Links complex to targets which make up the complex
COMPLX	CMPHD	COMPTG	Links complex header to complexes
CONTRY	ASNTAB	ASNCTY	Links assignment table header to assigned countries
DEPCOR	DPCHD	DEPCRD	Links depenetration corridor header to depenetration corridors
DEPDST*	TARCDE	TDDIST	Links target to distance to depenetration corridors
DEPLEG	DEPCRD	DPCRLG	Links depenetration corridors to its doglegs
DESTYP*	ASNDES	TYPDES	Links assigned DESIG to link to assigned type
DETOKE*	DEPCRD	RDDIST	Links depenetration corridor to distance to recovery base

*Linked to master

Table 11. (Part 2 of 4)

CHAIN NAME	MASTER RECORD	DETAIL RECORD	DESCRIPTION
DETOTG*	DEPCRD	TDDIST	Links depenetration corridor to distance to target
DSPITM	DISPRC	DISPDT	Links display table to its elements
DSPLAY	DISPHD	DISPRC	Links display header to display tables
ELEMNT	PRMADV	ADVELM	Links elemental adverb to its legal elements
EQUATE	PAYTBL	FOOTEQ	Links footprint equations to the proper payload table
EVENT	SRTYTB	SRTEVA	Links sortie table to event type A (weapon event)
EVENT	SRTYTB	SRTEVB	Links sortie table to event type B (non-weapon event)
GRPREG	RGION	WEPNGP	Links region to weapon groups in that region
IALINK*	INDRCT	LINKER	Links record type to link to attributes
IRDET	INDRCT	INDDET	Links record type to record showing chains of which it is a detail
IRMAST	INDRCT	INDMST	Links record type to record showing chains of which it is master
LISTXX	TARNUM	TARCDE	Links target list header to elements of the list
METOTG*	PENCRD	TPDIST	Links penetration corridor to distance to target
MYASGN*	WEPNGP	ASSIGN	Links weapon group to fixed assignments
MYBASE*	WEPSUB	MSBMTG	Links weapon subtype to missile/bomber targets that are its bases
MYEVNT*	ASSIGN	SRTEVA	Links weapon assignments to their sortie event
MYNAMZ*	INDRCT	INDHCS	Links record type for headers to their reference codes
MYPAY	PAYTBL	PYLDCT	Links payload table to warhead type count

*Linked to master

Table 11. (Part 3 of 4)

CHAIN NAME	MASTER RECORD	DETAIL RECORD	DESCRIPTION
MYSQDN*	WEPNGP	MSBMTG	Links weapon group to missile/bomber targets which provide bases for the group
MYSRTY*	WEPNGP	SRTYTB	Links weapon group to its sortie tables
NAMEZ	INDTHD	INDHCS	Links index header to header reference codes
PAYTAB	PAYHD	PAYTBL	Links payload table header to payload tables
PAYWEP*	PAYTBL	WEPSUB	Links payload table to weapon sub-type that uses it
PENCOR	PNCHD	PENCRD	Links penetration corridor header to penetration corridors
PENDST*	TARCDE	TPDIST	Links target to distance to penetration corridor
PENLEG	PENCRD	PNCRLG	Links penetration corridor to its doglegs
RCTYP	INDTHD	INDRCT	Links index header to record types
RECDST*	RECBTG	RDDIST	Links recovery base to distance to depenetration corridors
RECOVB	RCBHD	RECBTG	Links recovery base header to recovery bases
REFREG	RGION	REFPNT	Links region to refuel points in the region
REFUEL	RFPTHD	REFPNT	Links refuel point header to refuel points
REGION	REGHD	RGION	Links region header to regions
SORTIE	SRTYHD	SRTYTB	Links sortie header to sortie tables
TAB	DCTHD	DCTTAB	Links dictionary header to its tab characters
TABXYZ	TABLST	TABLEZ	Links utility table header to utility tables
TARGXX*	TARGET	MSBMTG	Links target to missile/bomber target additional data
TARGXX*	TARGET	RECBTG	Links target to recovery base additional data

*Linked to master

Table 11. (Part 4 of 4)

CHAIN NAME	MASTER RECORD	DETAIL RECORD	DESCRIPTION
TGTREG*	RGION	TARGET	Links region to targets in region
TGTTGT*	TARGTY	TARGET	Links target type to targets of that type
TGTTYP*	TGTHD	TARGTY	Links target header to target types
TYPRNG	ASNTYP	ASNREC	Links assigned type to category range for that type
VALIST	INDATR	ALPHVL	Links 'list' attribute to its legal values
VERB	SYNHD	SYNVRB	Links syntax header to verbs
WARHED*	WARHD	WRHEAD	Links warhead header to warhead types
WEPGRP	WPGPHD	WEPNGP	Links weapon group header to weapon groups
WEPNST	WEAPON	WEPSUB	Links weapon type to weapon subtype
WEPPAY*	WRHEAD	PYLDCT	Links warhead type to count in payload table
WEPTYP	WEPHD	WEAPON	Links weapon header to weapon types
WORD	DCTTAB	DCTWRD	Links tab character to words with that tab
WPINGP*	PAYTABL	WEPNGP	Links payload table to weapon groups using that table

*Linked to master

Table 12. Chains Which are Linked to Master

CHAIN NAME	MASTER RECORD	DETAIL RECORD
ADVERB	PRMADV	SYNCLZ
AILINK	INDATR	LINKER
ALTDES	ASNTYP	TYPDES
ASGWPN	TARCDE	ASSIGN
ASNRNG	ASNCTY	ASNREC
CMPTGT	COMPTG	TARGET
DEPDST	TARCDE	TDDIST
DETORE	DEPCRD	RDDIST
DETOTG	DEPCRD	TDDIST
DESTYP	ASNDES	TYPDES
IALINK	INDRCT	LINKER
METOGP	PENCRD	GPDIST
METOTG	PENCRD	TPDIST
MYASGN	WEPNGP	ASSIGN
MYBASE	WEPSUB	MSBMTG
MYEVNT	ASSIGN	SRTEVA
MYNAMZ	INDRCT	INDHCS
MYSQDN	WEPNGP	MSBMTG
MYSRTY	WEPNGP	SRTYTB
PAYWEP	PAYTBL	WEPSUB
PENDST	TARCDE	TPDIST
RECDST	RECBTG	RDDIST
TARGXX	TARGET	MSBMTG
TARGXX	TARGET	RECBTG
TGTREG	RGION	TARGET
TGTTYP	TARGTY	TARGET
TGTTGT	TARGTY	TARGET
WARHED	WARHD	WRHEAD
WEPPAY	WRHEAD	PYLDCT
WPINGP	PAYTBL	WEPNGP

Table 14. Internal COP Common Block

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
C25		Represents record type INDHCS. Each record is used to identify the BCD reference code of a header.
	XCLASS	Header's CLASS value
	XSIDE	Header's SIDE value
	XREFCD	Header's BCD reference code (this variable is type character *8)
C35	LINKS(100)	Module Link Table
FIRST	INCOM	Logical switch which, when on, indicates a sentence is being analyzed
SYMBOL	KYMBOL	Used to pass constructed ERRFND symbol (see section 3.5.2).
TABLZ		Contains buffers for utility tables used to store ERRFND table
	KTBVAL(100,4)	Buffer of 100 words for each table type
	TBRFCD(50,4)	Contains Reference Codes for tables
	NUMOT(4)	Number of utility tables created
	NUMCT(4)	Index numbers of table currently in buffer.
VBINDX	VIND	Number of data transaction produced by COP which may be cross-referenced with the number of the data transaction listed in the data module quality control list
	NVB	Integer variable used to increment VIND
	ICRSW	Integer switch used to save first value of VIND into HAREA
	HAREA	Character variable used to suppress printing of duplicate cross-reference numbers.

3.7 Main Routine of COP

PURPOSE: Main program of executive module

ENTRY POINTS: COP (for purpose of discussion)

FORMAL PARAMETERS: None

COMMON BLOCKS: C15, IPQT, OOPS

SUBROUTINES CALLED: CLZIDS, DELTAB, ERRFND, INICOP, INPTRN, INSDEL, MODGET

CALLED BY: HIS operating system

Method:

First, several switches are set: CHCKOV to control a later call to DELTAB, ERROR to indicate no error has occurred yet, and ENDSW to indicate no end of input. Next, INICOP is called. The process which follows occurs for each command sentence.

First the ERRFND overlays are read in (this does not occur for the first sentence since INICOP has already done this). Next, ERRFND is called. If CHCKOV is still true, the INPTRN link is read in and INPTRN is executed if no error has occurred. If an error occurred before INPTRN, only DELTAB is called.

Next, if no error has occurred, MODGET is called to execute the module. If an error has occurred, CHCKOV is set to false.

Finally, the End Input switch is checked (ENDSW); if not on, the next input sentence is processed. If it is on, CLZIDS is called and processing stops.

COP is illustrated in figure 13.

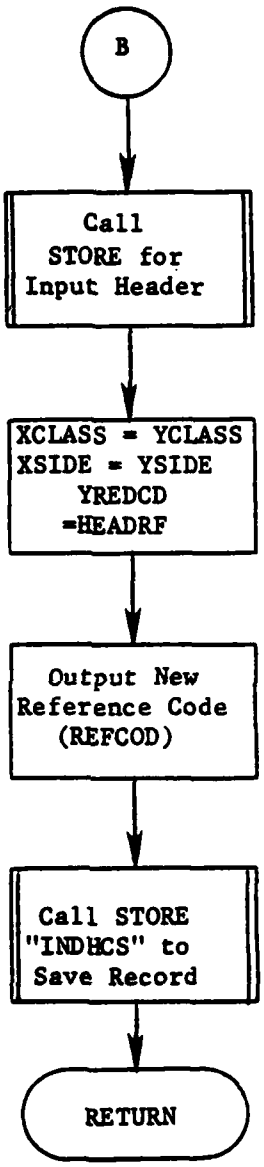


Figure 16. (Part 5 of 5)

3.7.4 Subroutine INICOP

PURPOSE: To initialize COP headers and check for special run modes

ENTRY POINTS: INICOP

FORMAL PARAMETERS: None

COMMON BLOCKS: C15, C30, ERRCOM, IPQT, OOPS, STRING, VBINDX

SUBROUTINES CALLED: BANNER, BOOT, DLETE, ENTMOD(SRM), ERPRIN, GETSTR, HDFND, OPNIDS, RETRV, STORE, WEBSTR

CALLED BY: COP

Method:

The standard header is produced and GETSTR called for the first string. If the first string of input is "RESTORE," the overlay for the Save and Restore Module (SRM) is executed, IDS file opened and GETSTR is called again. The current string is now checked for the value "INITIALIZE." If this is the value, the overlay for BOOT is executed.

In any case, the next string is obtained from GETSTR and the utility tables purged. Finally, various headers are retrieved and the syntax analysis process begun by bringing in the ERRFND overlay and calling WEBSTR.

Subroutine INICOP is illustrated in figure 17.

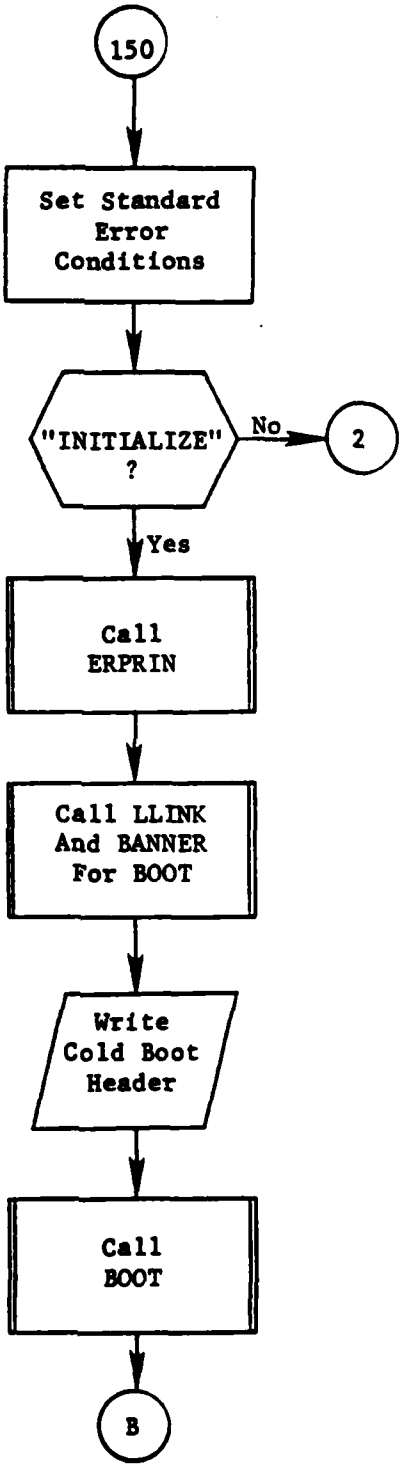


Figure 17. (Part 3 of 4)

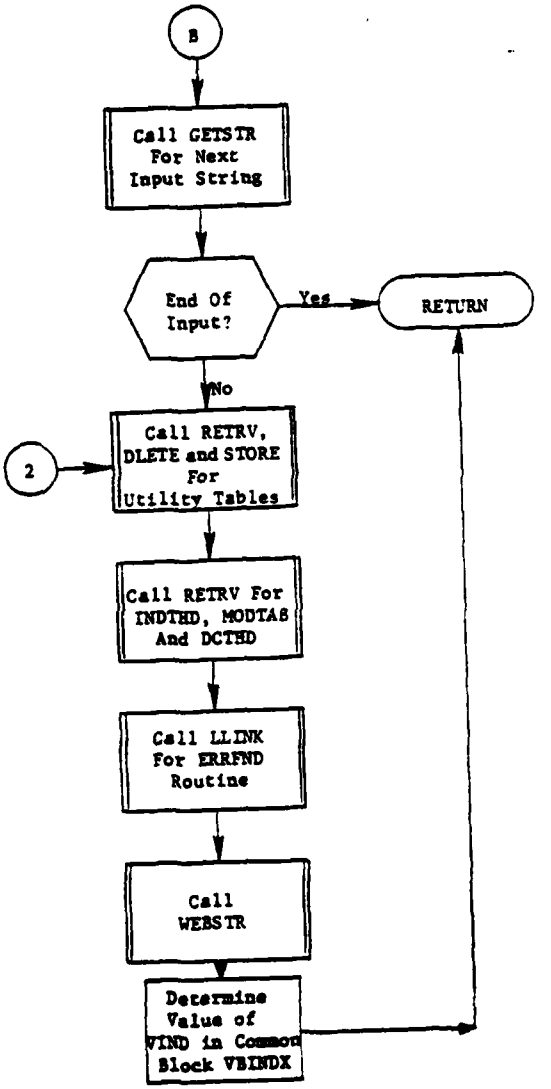


Figure 17. (Part 4 of 4)

3.7.4.1 Subroutine INPRIN

PURPOSE: Control input print

ENTRY POINTS: INPRIN, ERPRIN, LGPRIN

FORMAL PARAMETERS: None

COMMON BLOCKS: IPQT, VBINDX

SUBROUTINES CALLED: None

CALLED BY: COP, GETSTR, INICOP, LNGSTR, SYNTAX

Method:

Both the INPRIN and LGPRIN entry points have a similar process. The INPSW switch is checked and if false the contents of HOLD and HFLAG are printed (written to file 11). Then the current input line (INBUF) is stored in HOLD. HFLAG is set according to the entry point - blank for INPRIN, "*" for LGPRIN.

The ERPRIN entry point prints HOLD if it has not been printed and resets INPSW.

Subroutine INPRIN is illustrated in figure 17.1.

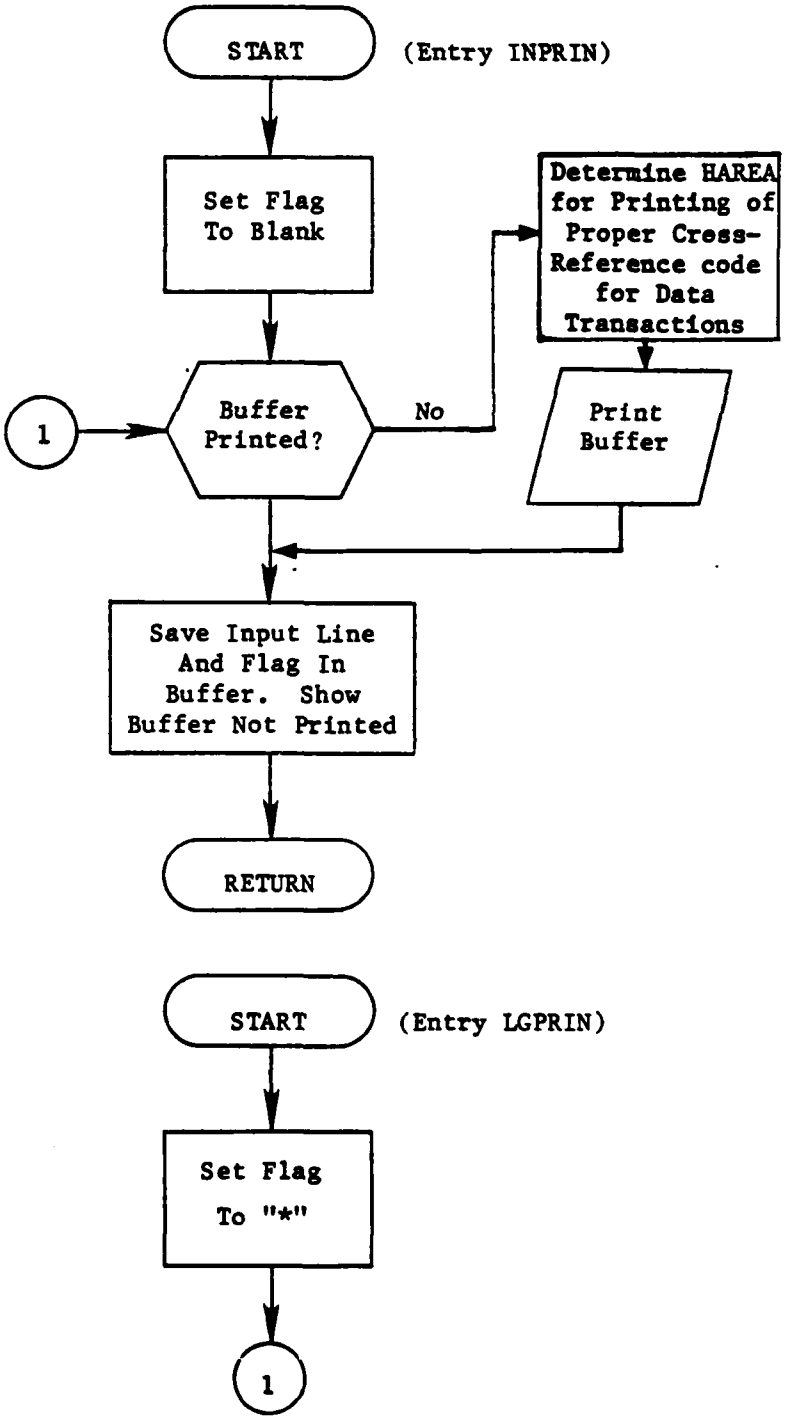


Figure 17.1. Subroutine INPRIN (Part 1 of 2)

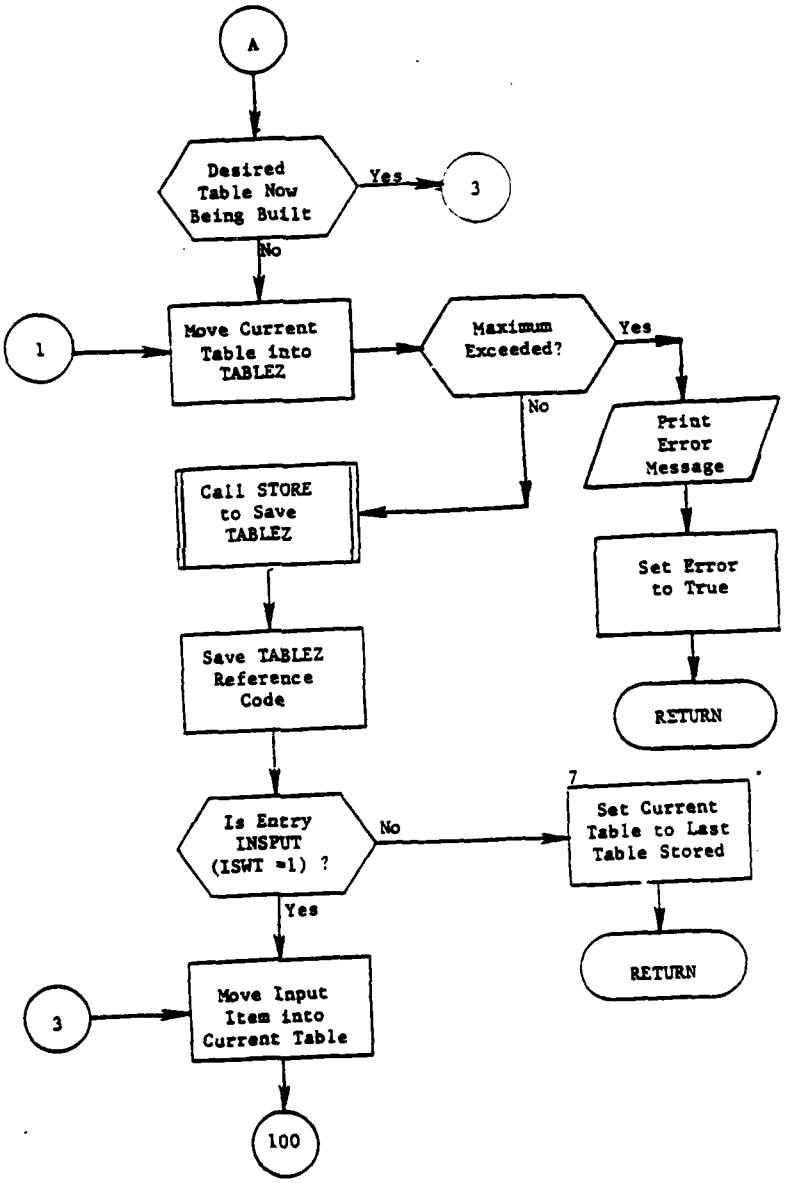


Figure 18. (Part 2 of 7)

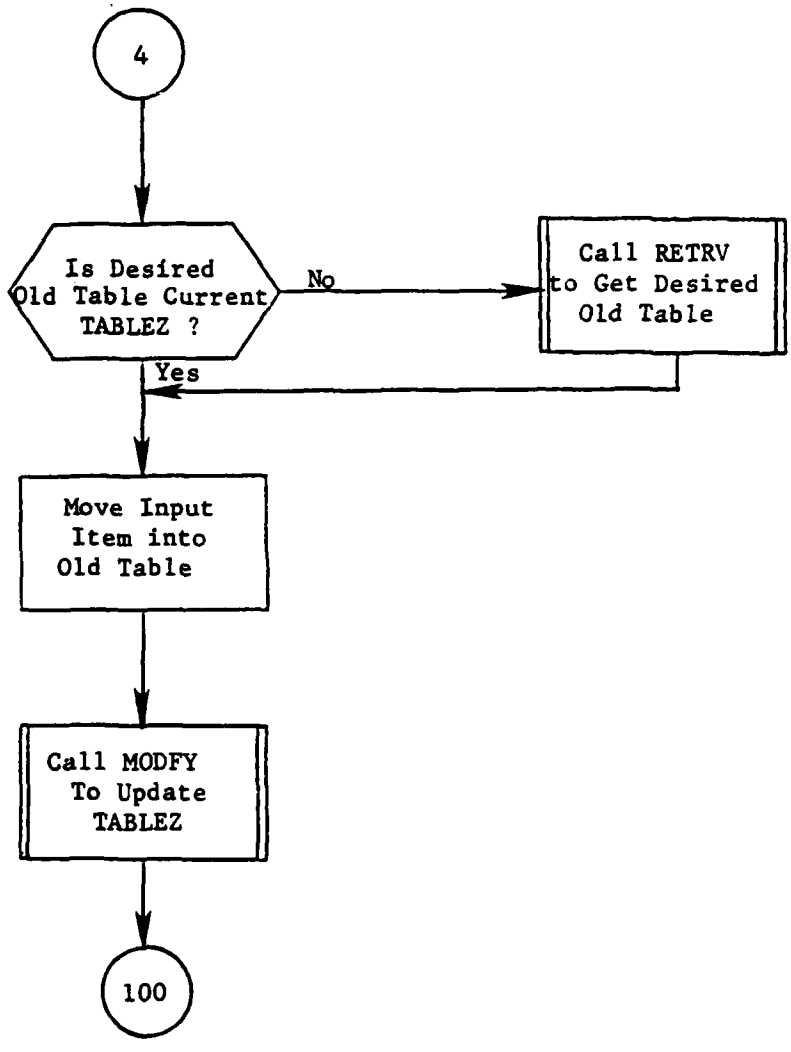


Figure 18. (Part 3 of 7)

3.7.6 Subroutine MODGET

PURPOSE: Execute modules

ENTRY POINTS: MODGET

FORMAL PARAMETERS: None

COMMON BLOCKS: C35, ERRCOM, OOPS, QC

SUBROUTINES CALLED: BANNER, ENTMOD, INSGET

CALLED BY: COP

Method:

First INSGET is called to get the verb's number. This number is used as an index to the module link table (common block C35) to obtain an overlay link name. This name (NEWMOD) is compared to the old name (OLDMOD). If they are different, BANNER is called to display NEWMOD. OLDMOD is set to NEWMOD. System routine LLINK is called to read in overlay NEWMOD and standard module entry point ENTMOD is called. The standard error conditions are now reset. If an error occurred during module execution an error message is produced. Finally, if either the DATA or REPORT module was executed, the error flag is reset to false. If the data module is called, the heading for the data module quality control is written to file code 13.

Subroutine MODGET is illustrated in figure 19.

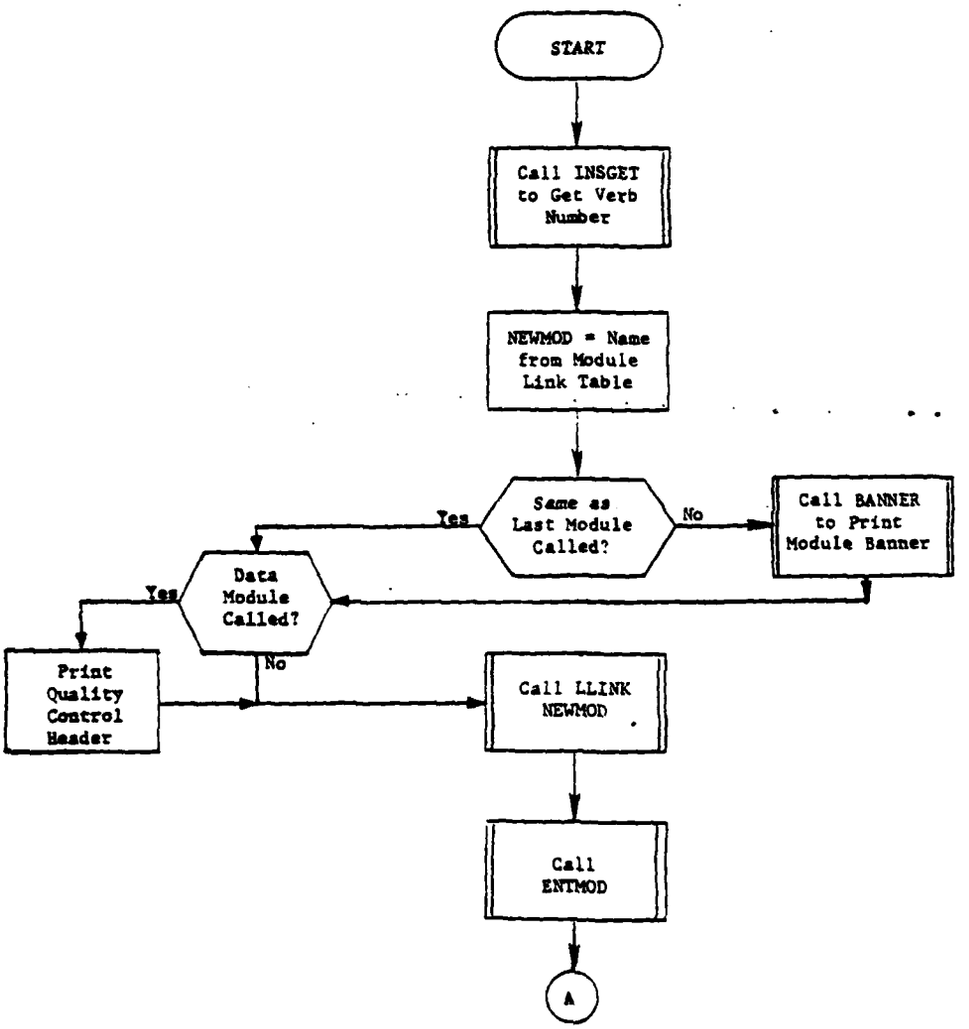


Figure 19. Subroutine MODGET (Part 1 of 2)

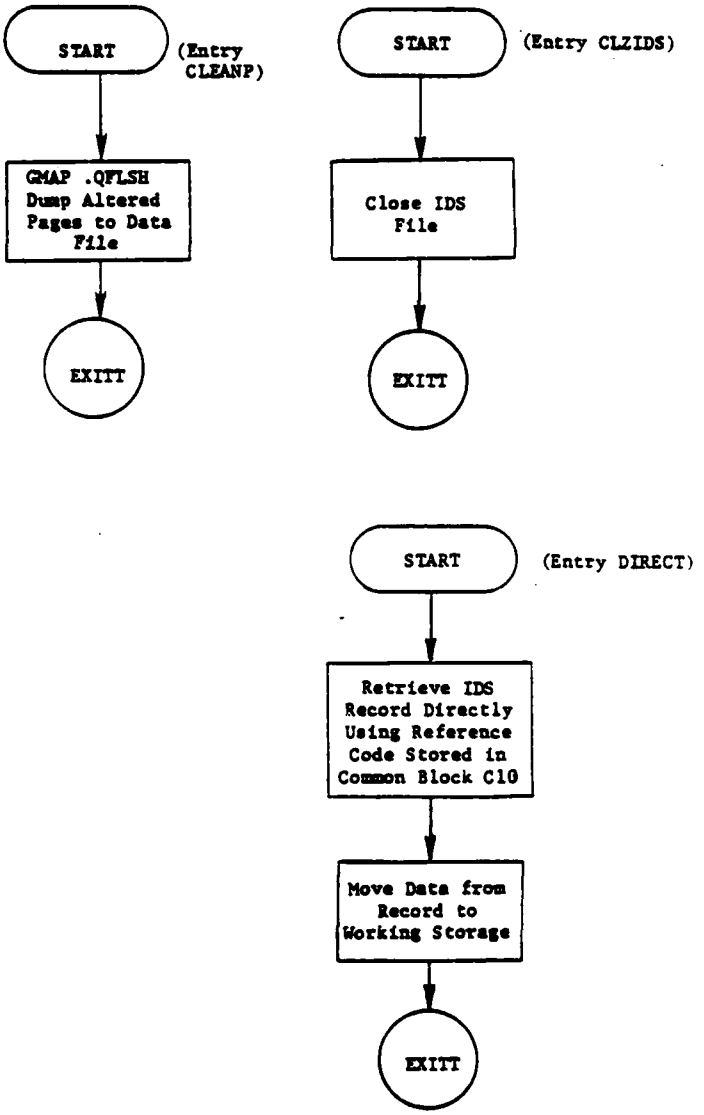
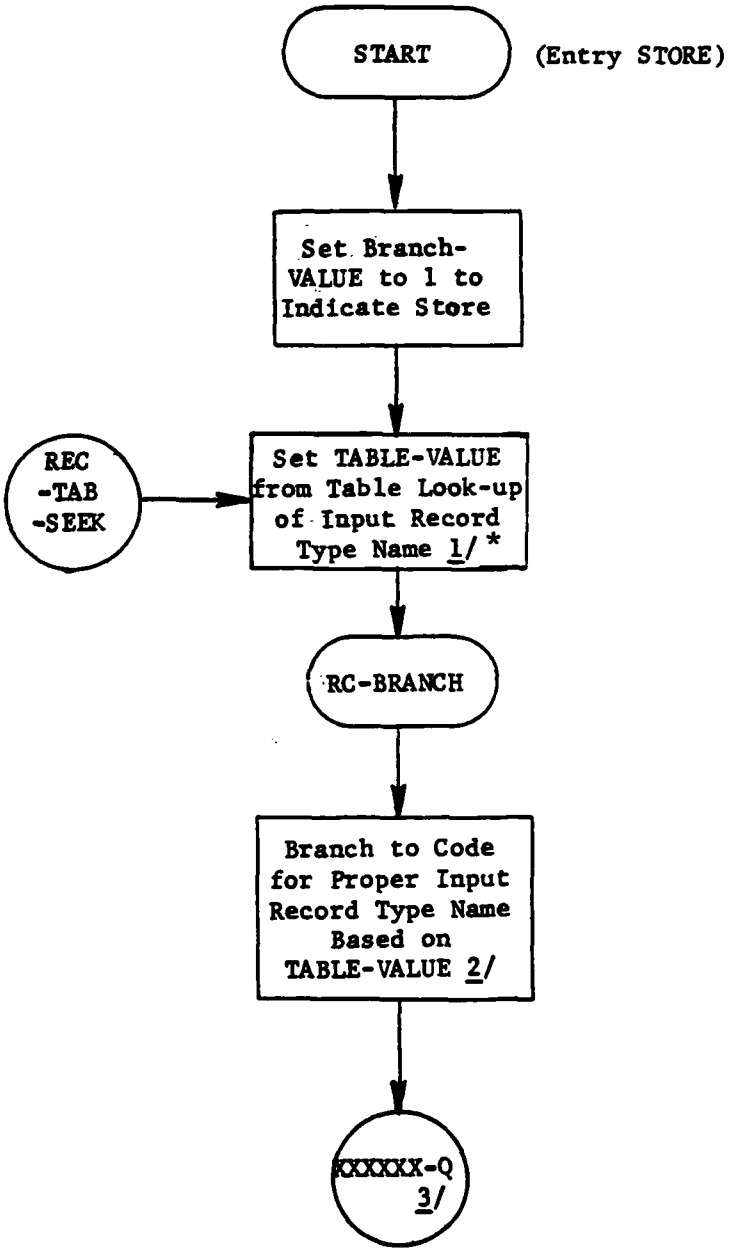


Figure 20. Subroutine QDATA: Entry CLZIDS and DIRECT (Part 2 of 9)



* See part 9 for explanation of annotated notes

Figure 20. Subroutine QDATA: Entry STORE (Part 3 of 9)

3.8 Subroutine BOOT*

PURPOSE: Create and update organizational data

ENTRY POINTS: BOOT

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C15, C20, C30, C35

SUBROUTINES CALLED: DCTFND, HDFND, HDPUT, HEAD, MNMFND, MODFY, NEXTTT, NUMFND, RETRV, RNMFND, SEEKER, STORE, STRMAK

CALLED BY: INICOP

Method:

BOOT reads card images which instruct it as to what actions to take. The card images are in sets which are begun by an introductory adverb and ended with an END card. The first part of theprocess is to read an adverb (NEWINDEX, RECORDTYP, INDEX, DICTIONARY, SYNTAX, MODULE, HEADER). Each adverb causes the branch IBR to be set to a different value. If an adverb is read which is not recognized processing ceases. The method used for each adverb is different. However, each card image that is read is printed after any action is taken with an appropriate flag (IND).

NEWINDEX

No command cards follow this abverb. The action taken is to set the CLASS attribute and call STORE to create the utility table and index headers.

RECORDTYP

Each card image creates a new INDRCT record. The process is to call NUMFND for the record type number then search the RCTYP chain for a match. If a match is found, IND is set for ignored input. If no match is found, IND is set to show added record and STORE is called to create an INDRCT record.

INDEX

Each card image creates a new record of the type specified in the first field of the card image.

*Main routine of overlay BOOTT.

INDMST or INDDET: RNMFND is called for the record type numbers and to check the validity of both record type names. SEEKER is then called on the appropriate chain (IRMAST or IRDET, respectively) to look for a duplicate. If a duplicate is found the ignored flag (IND) is set. Otherwise, the appropriate attributes are set and STORE is called.

INDATR: First STRMAK is called to obtain the attribute name. Then this name is checked for validity by DCTFND. Next the ATRIB chain is queried by SEEKER. If a match is found IND is set to indicate a change, if not it is set to indicate an add. Now MNMFND is used to get the value of the attribute type. This type is used to determine how to read the default and range fields. Finally, depending upon IND, either STORE or MODFY is called.

ALPHVL

First STRMAK is called to obtain the attribute name, and the name is validated by DCTFND. Next SEEKER is used to find this attribute on the ATRIB chain. Then SEEKER is used to check the VALIST chain for a duplicate value. Finally, if all checks are correct, STORE is called to add a ALPHVL record.

LINKER

First STRMAK is called to obtain the attribute name and it is validated by DCTFND. Next RNMFND is used to validate the record type name. Next the IALINK chain is searched by SEEKER for a duplicate, which, if found, causes the flag (IND) to be set for a change. Then MNMFND is called to obtain the value of the control mnemonic. Finally, either MODFY or STORE is called depending upon the flag (IND).

DICTIONARY

Each card image creates a new entry in the dictionary. First STRMAK is called to obtain the input value for the word to be entered in the dictionary. Next DCTFND is called to look for the new word in the dictionary. If the word is found the indicator flag (IND) is set for a change. If neither the word nor its tab character (tab character is formed from the first two characters of the word) is found, STORE is called to create an appropriate tab character record (DCTTAB). Now MNMFND is called to set the word type. If the type is attribute (Type=6) NUMFND is called for the value, and the address and MNMFND is called for the type and identifier flag. These quantities are packed into WORDVL. If not an attribute, NUMFND is called for WORDVL. Finally, either MODFY or STORE is called.

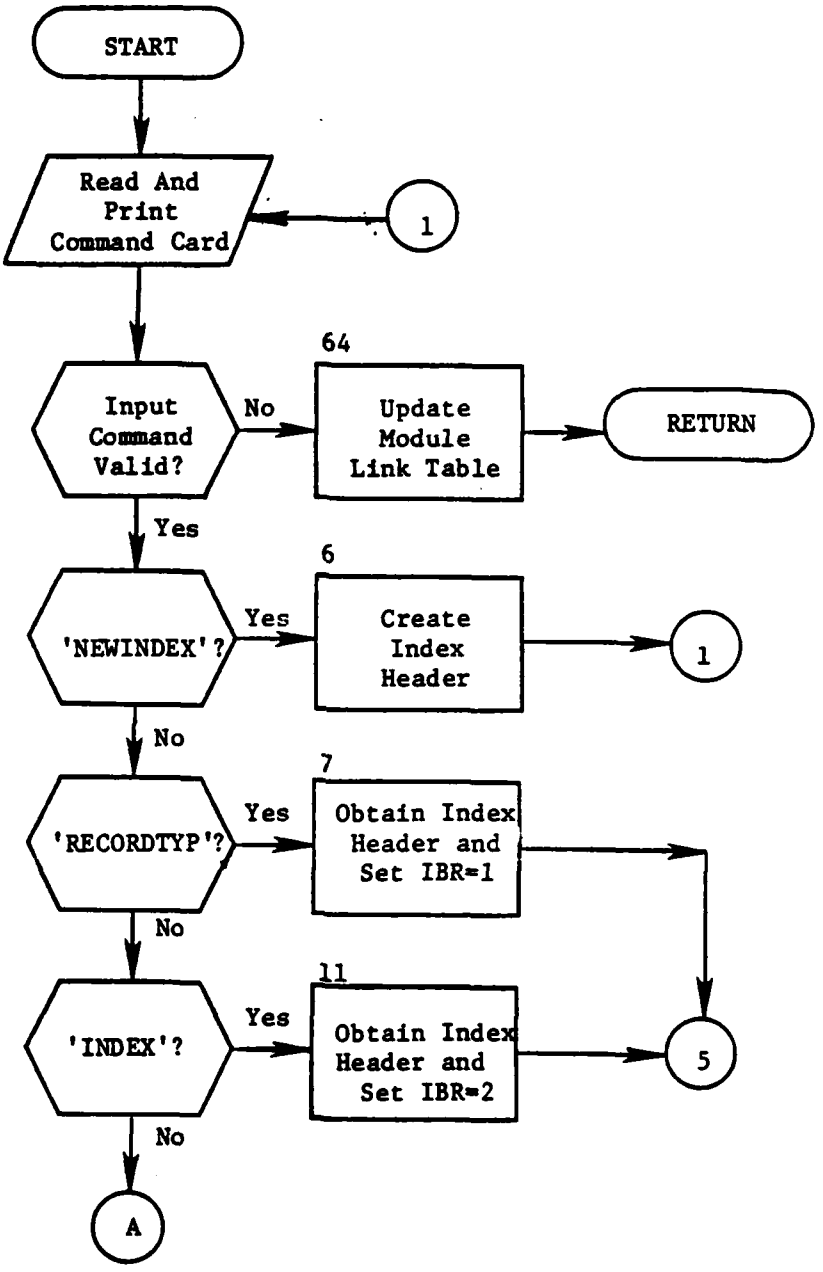


Figure 21. Subroutine BOOT (Part 1 of 24)

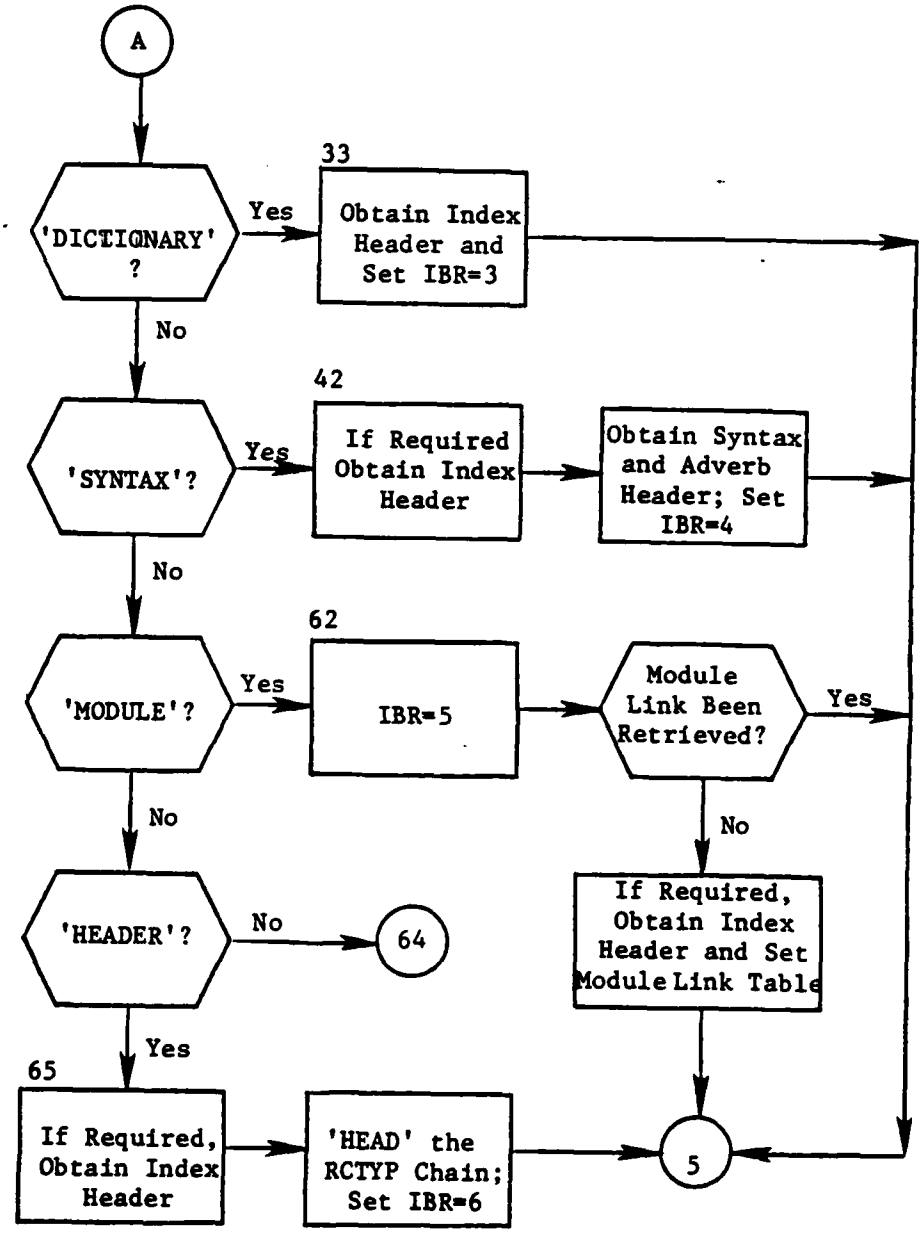


Figure 21. (Part 2 of 24)

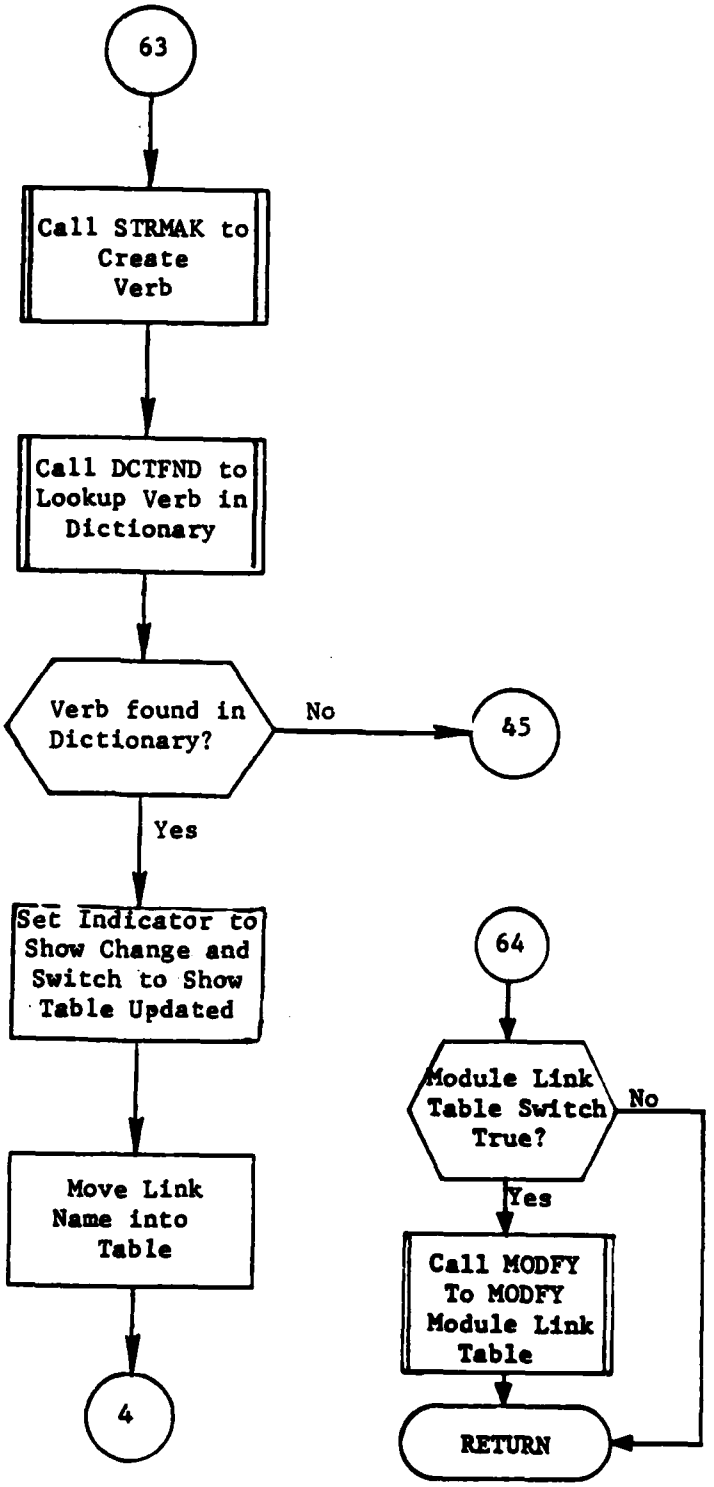


Figure 21. (Part 23 of 24)

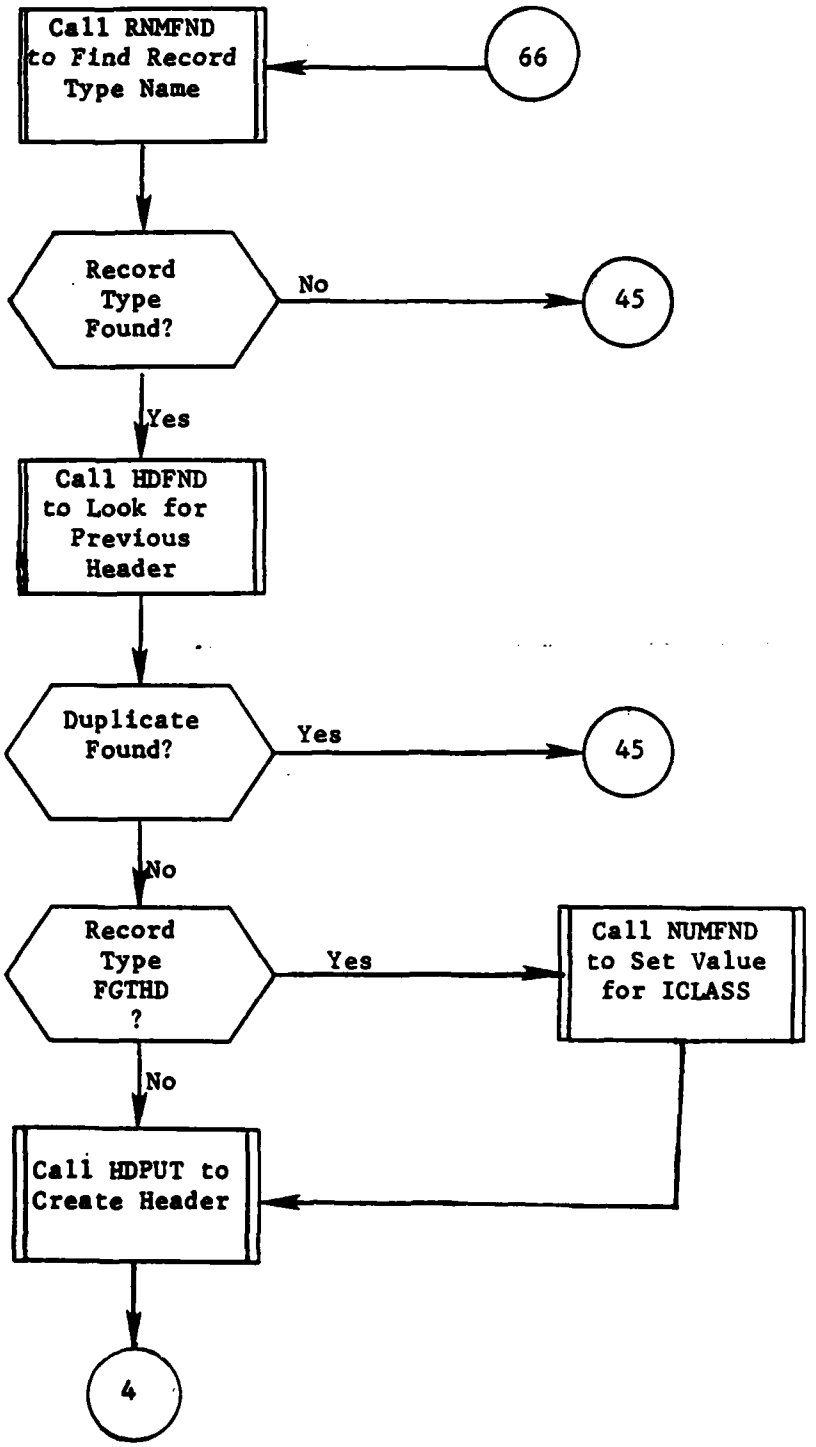


Figure 21. (Part 24 of 24)

3.8.1 Subroutine DCTFND

PURPOSE: Check for dictionary match

ENTRY POINTS: DCTFND

FORMAL PARAMETERS:
STRING: Character string to be searched for
TYPE: Type of string expected
NUMBER: Identifying number returned
ADRESS: Address of attribute returned

COMMON BLOCKS: C10, C15, C30

SUBROUTINES CALLED: HDFND, NEXTTT, RETRV

CALLED BY: BOOT

Method:

First the HDSAVE variable is checked. If blank HDFND is called to set the value of the reference code of the dictionary header into HDSAVE. The dictionary header is retrieved. ICOMP is set to the first two characters of the input string to be used as a tab character. The TAB chain is now searched to find a match for ICOMP. If not found, NUMBER is set to zero and the subroutine returns.

If a match is found for ICOMP, the WORD chain is searched for a match. If none is found, the subroutine returns with NUMBER=0. If a match is found the type of the word found is compared to TXPE. If no match, NUMBER is set to zero. If a match, NUMBER is set from the dictionary and, if the type is attribute, ADRESS is also set.

Subroutine DCTFND is illustrated in figure 22.

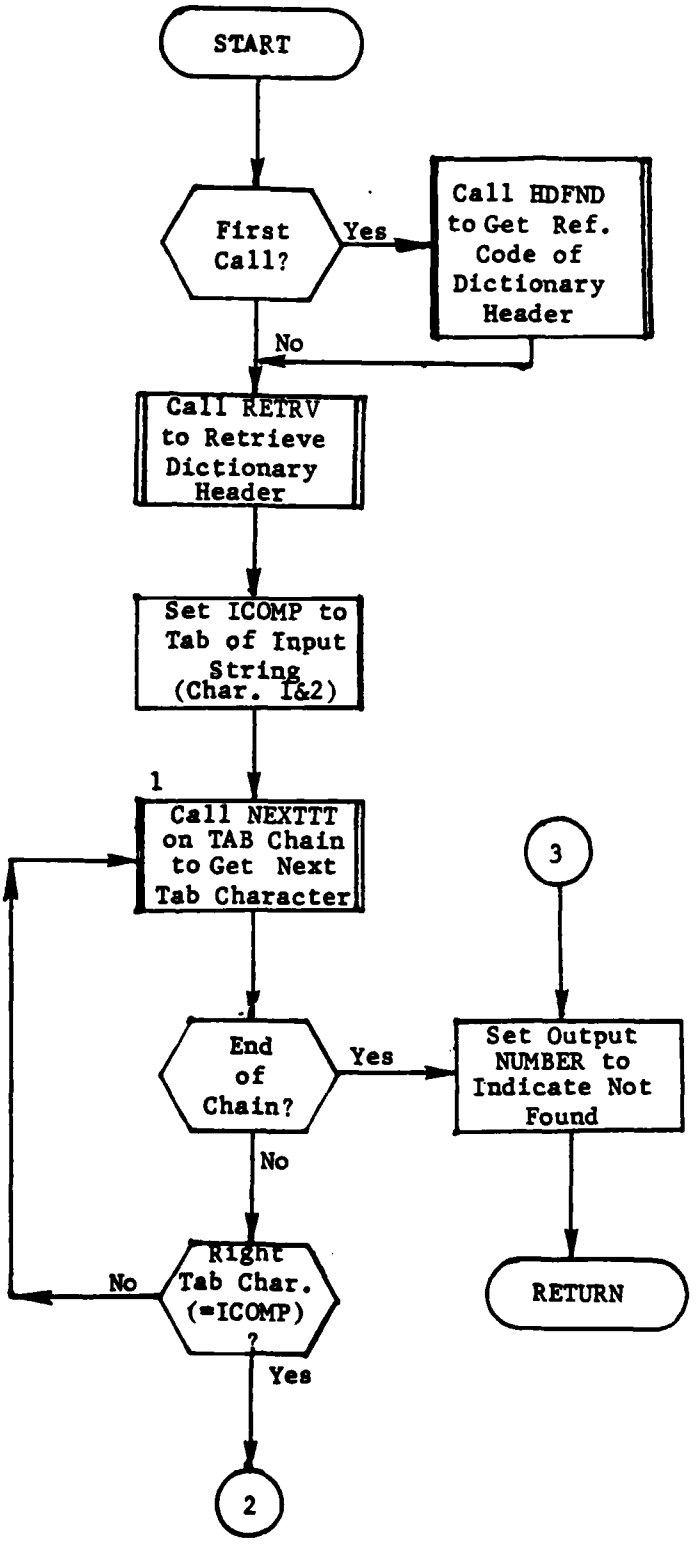


Figure 22. Subroutine DCTFND (Part 1 of 2)

3.9 Subroutine ERRFND*

PURPOSE: Control syntax analysis process

ENTRY POINTS: ERRFND

FORMAL PARAMETERS: None

COMMON BLOCKS: FIRST, IPQT, STRING, SYMBOL, TABLZ, VBINDX

SUBROUTINES CALLED: GETSTR, LNGSTR, SYNTAX, TABINS, WEBSTR

CALLED BY: COP

Method:

First the ERRFND table counts are set to zero and the SPCIAL switch is set to 'False' to indicate that the previous symbol is not an operator. For every input string but the first, GETSTR and WEBSTR are called. (INICOP has already called them in the case of the first sentence and for every other sentence the verb has been read.) If the call to GETSTR encounters an end of input (ENDSW), WEBSTR is not called. If the input string is a null (Type=11), GETSTR is called again.

Whether the first string of the sentence or not, process arrives at statement 3 (see figure 28). If the type is a long string (Type=2) LNGSTR is called. Next SYNTAX is called to check for syntax errors. If end of input, process stops here. Otherwise, the symbol is begun (KYMBOL) by setting it to TYPE. Then the rest of the symbol is created according to its type. Operators, adverbs, special words and verbs (which are at the beginning of the sentence) have their identifier values added to ther symbol and TABINS is called to save the symbol. Long strings have already had their symbols created. Attributes, alphabetics and numerics have their values stored by one call to TABINS and their symbols stored by another.

After a string's symbol is stored, the process returns to call GETSTR for the next string until an end of input or a second verb.

Subroutine ERRFND is illustrated in figure 28.

*Main routine of overlay ERRF

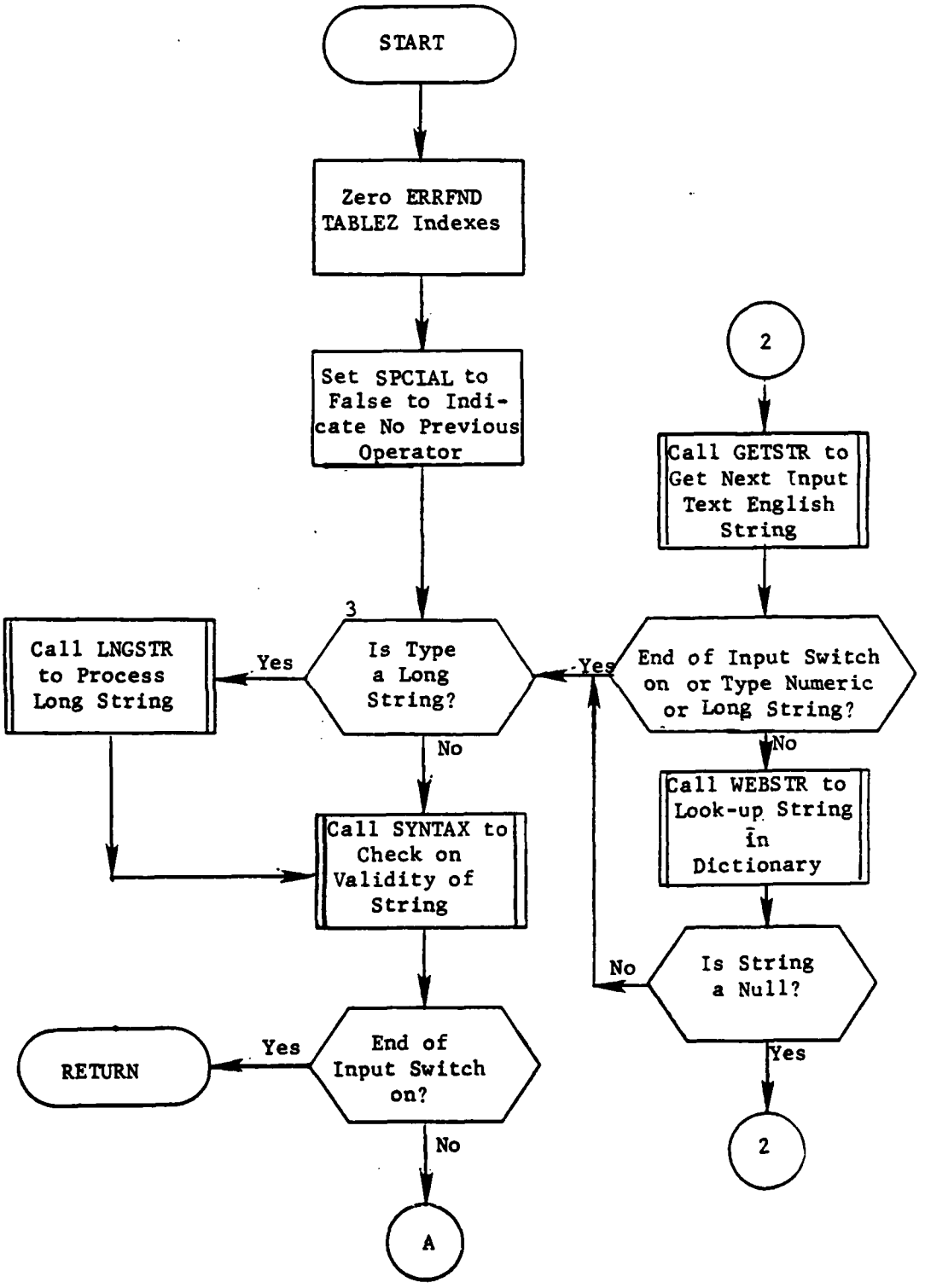


Figure 28. Subroutine ERRFND (Part 1 of 2)

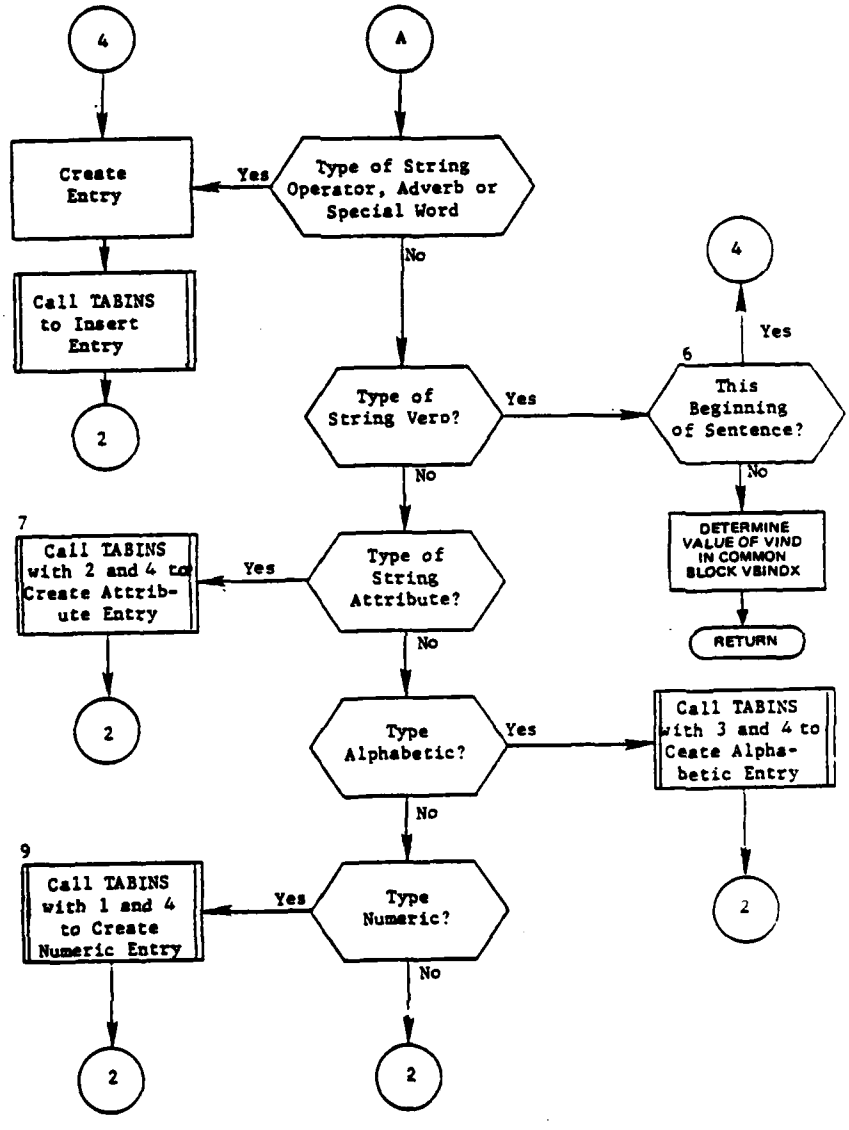


Figure 28. (Part 2 of 2)

3.9.1 Subroutine LNGSTR

PURPOSE: Processes long strings

ENTRY POINTS: LNGSTR

FORMAL PARAMETERS: None

COMMON BLOCKS: IPQT, STRING, SYMBOL

SUBROUTINES CALLED: TABINS, LGPRIN, ERPRIN

CALLED BY: ERRFND

Method:

The delimiter which caused the call is saved. Thereafter, the process continues to scan the input card image one character at a time until an identical character to the delimiter is encountered. With each character retrieved, the character count is incremented. If it exceeds 120, a warning message is printed and the remainder of the string ignored. Each character is added to the next position of ALPHA and if ALPHA is full it is stored in ALPHSV.

When all characters have been read, the length of the string is checked. If it is less than or equal to 12, the string is treated as an alphabetic, stored in ALPHA and TYPE is set to 9. Otherwise, a symbol containing 2 in bits 30-35 and the character count in bits 0-29 is stored by TABINS and an alphabetic constant and symbol are stored for each element of ALPHSV.

Subroutine LNGSTR is illustrated in figure 29.

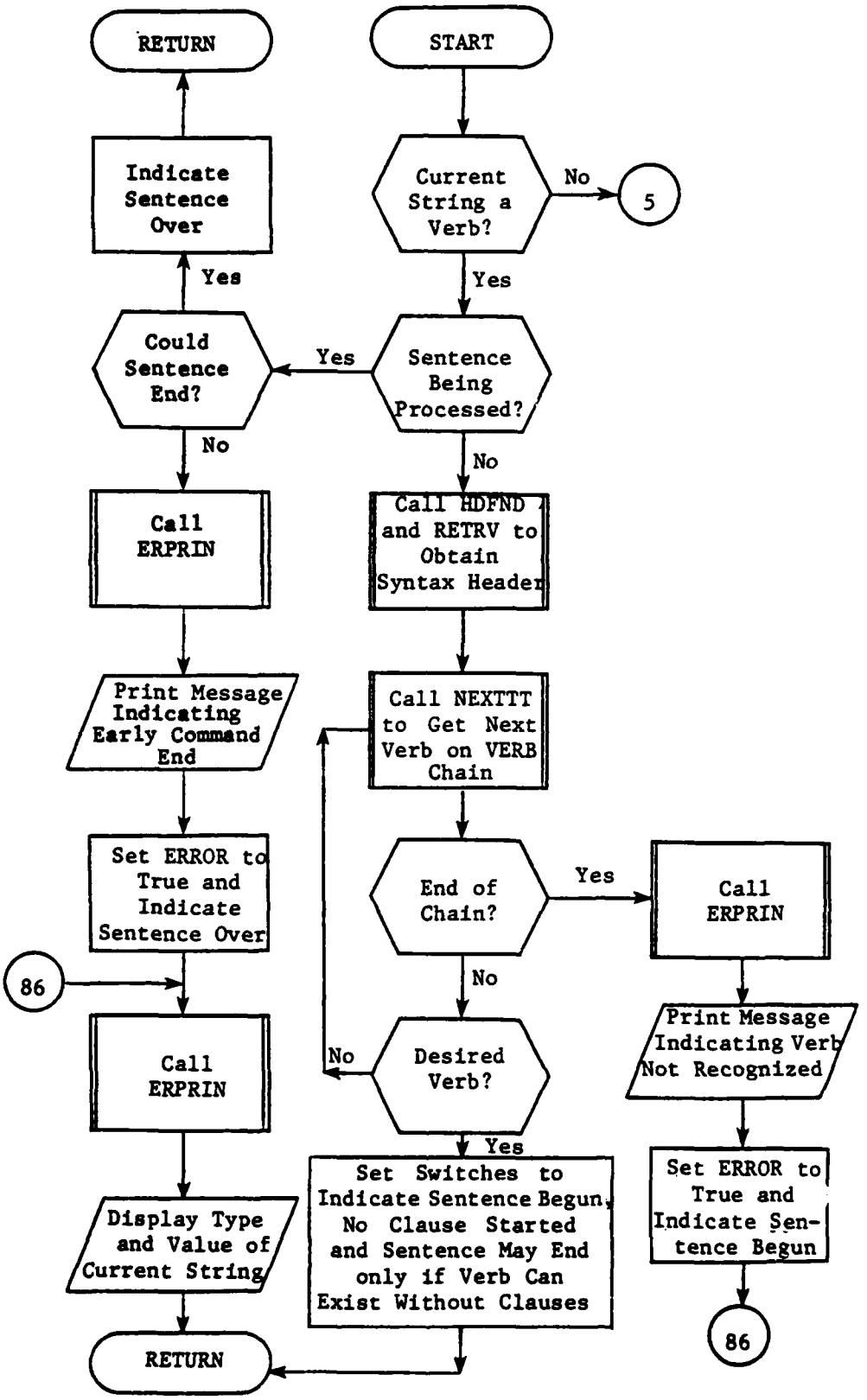


Figure 30. Subroutine SYNTAX (Part 1 of 22)

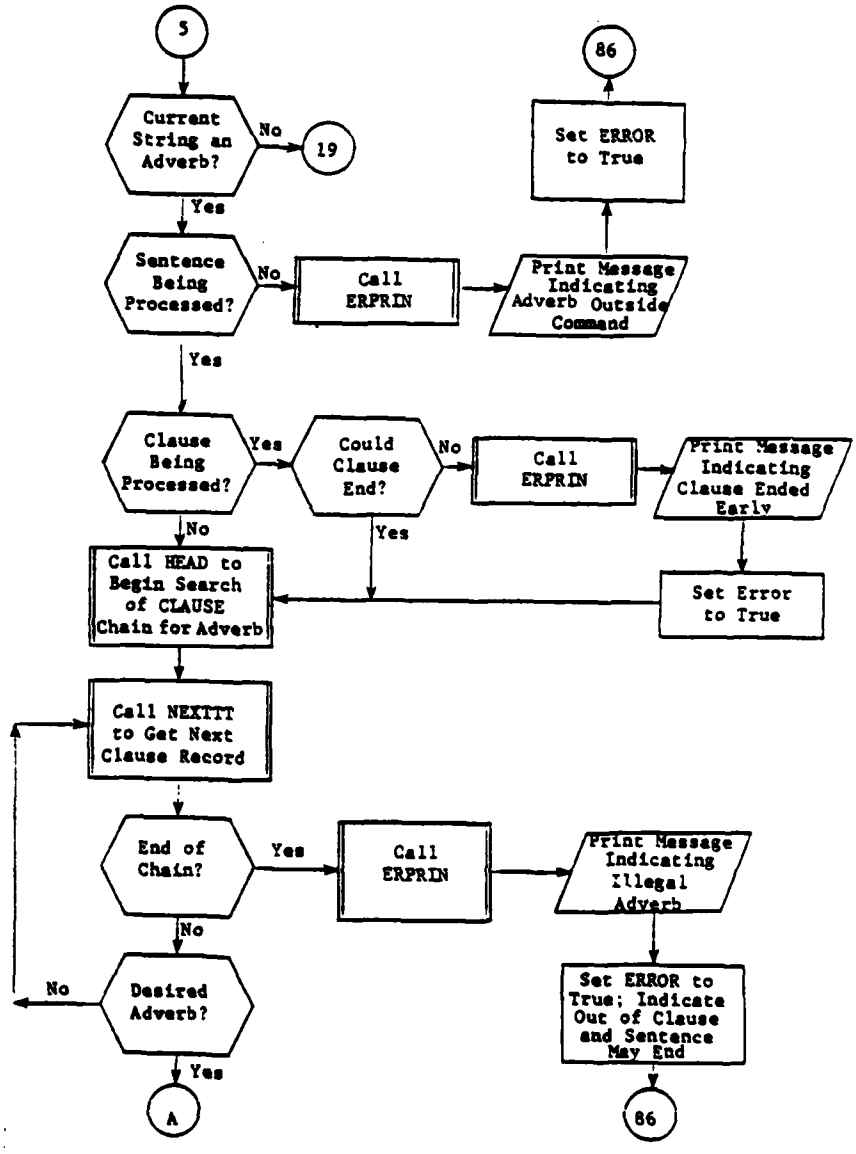


Figure 30. (Part 2 of 22)

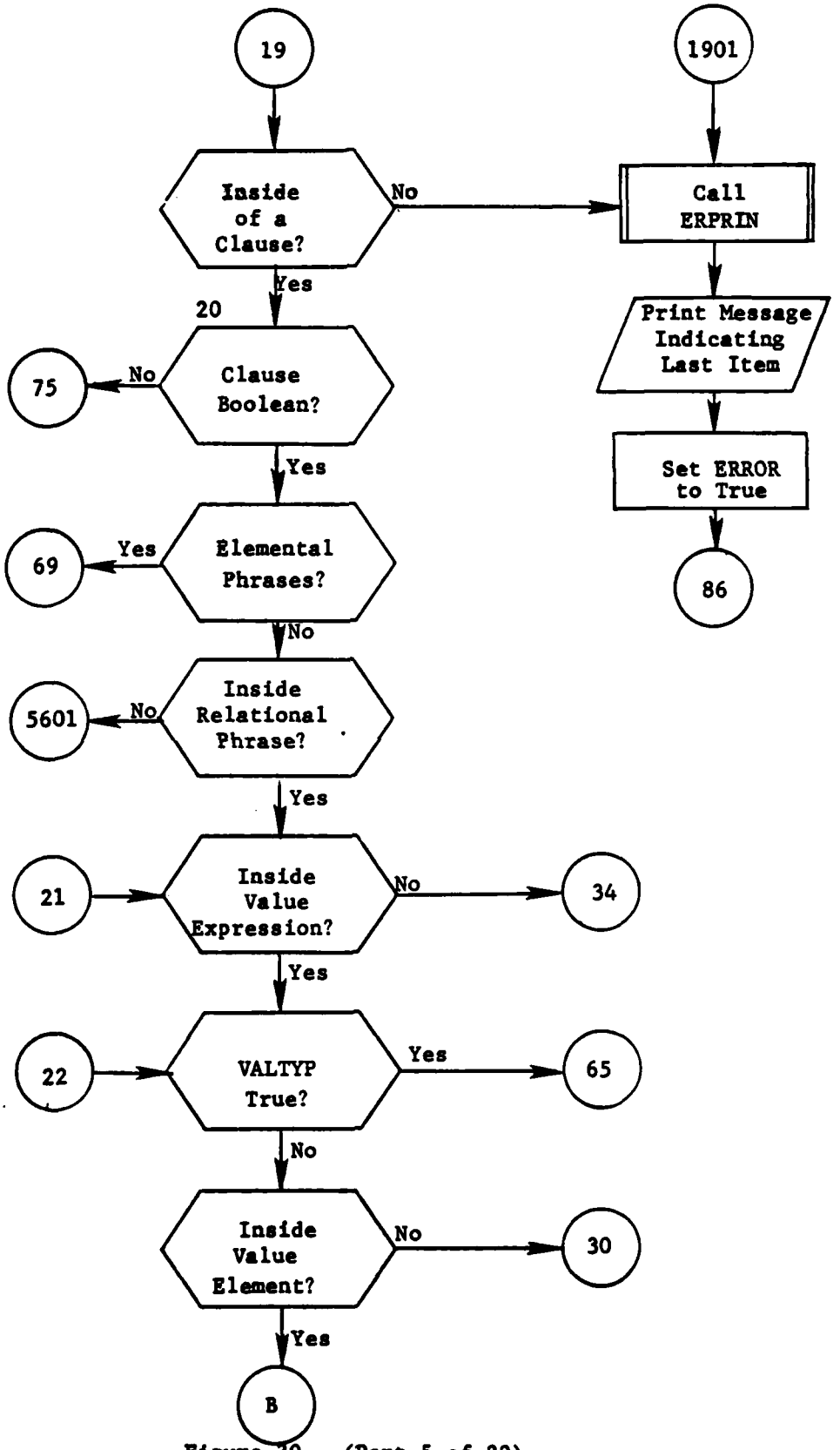


Figure 30. (Part 5 of 22)

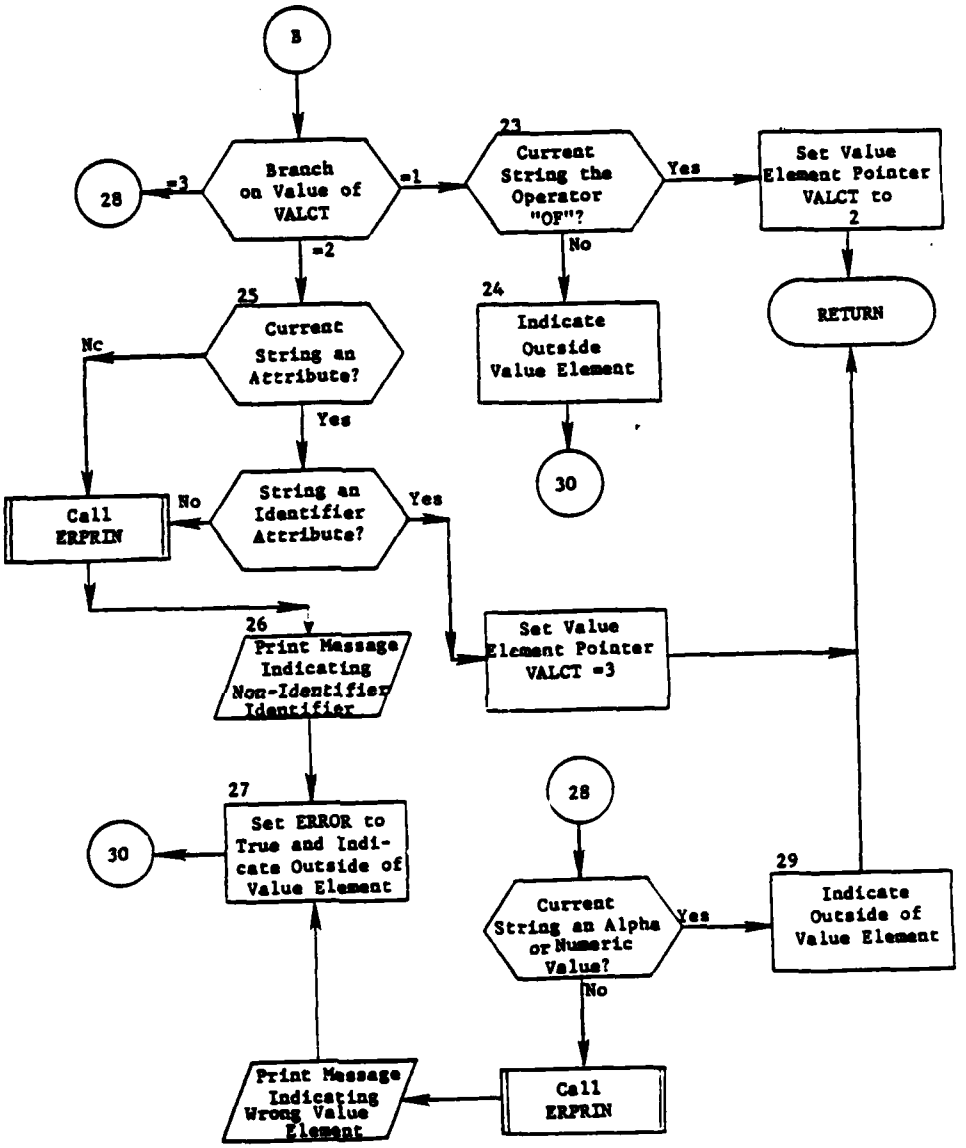


Figure 30, (Part 6 of 22)

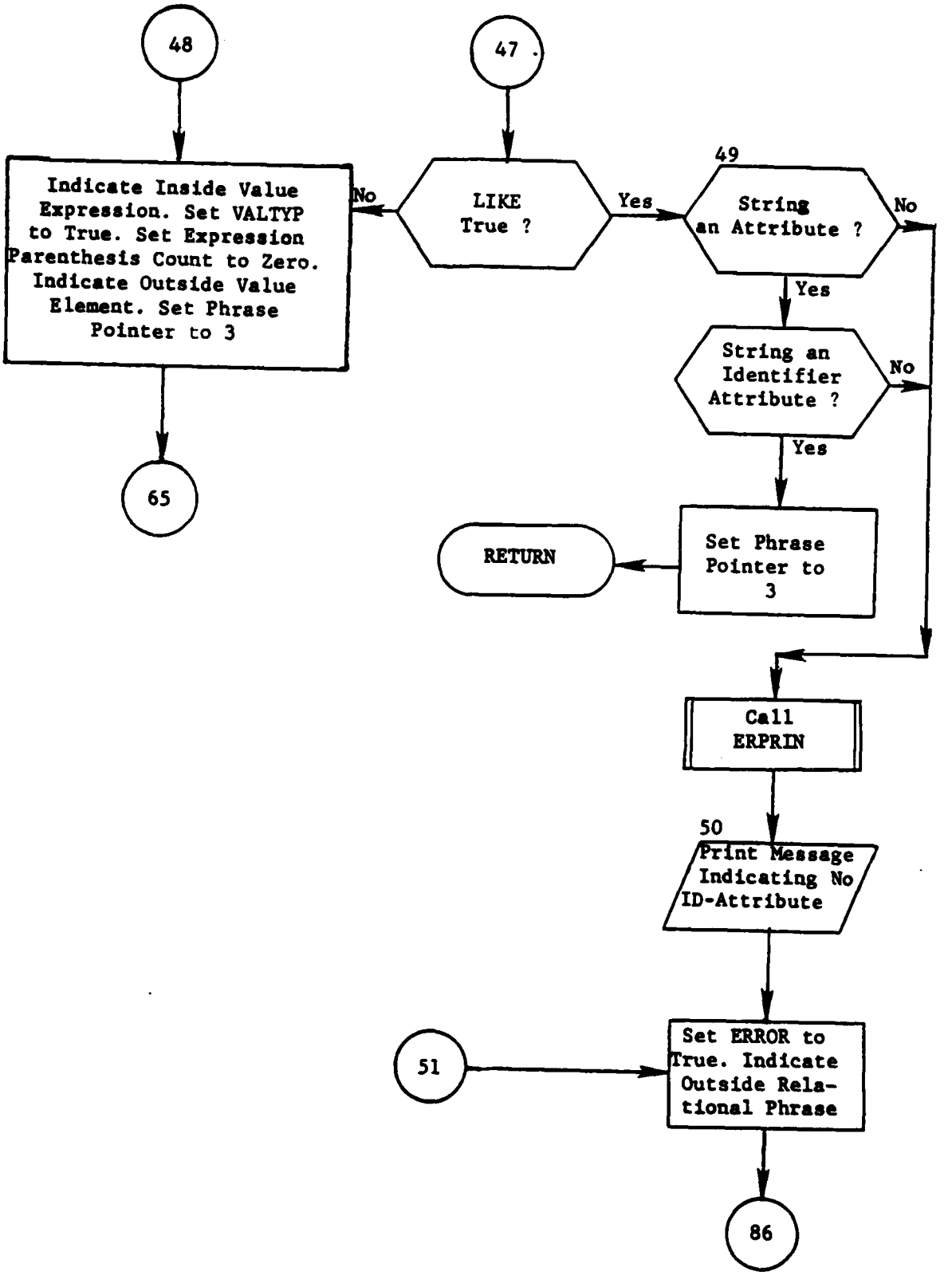


Figure 30. (Part 11 of 22)

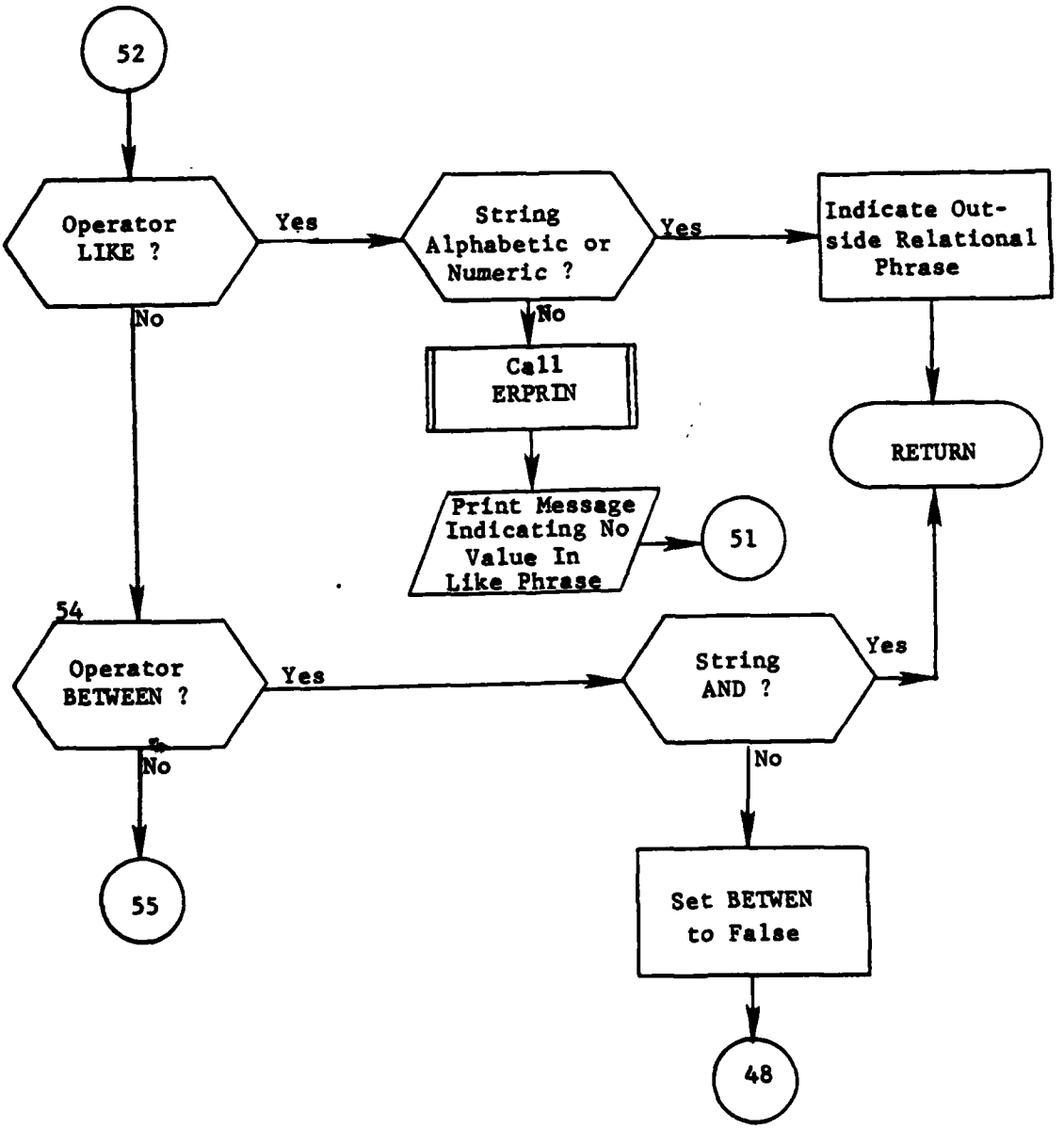


Figure 30. (Part 12 of 22)

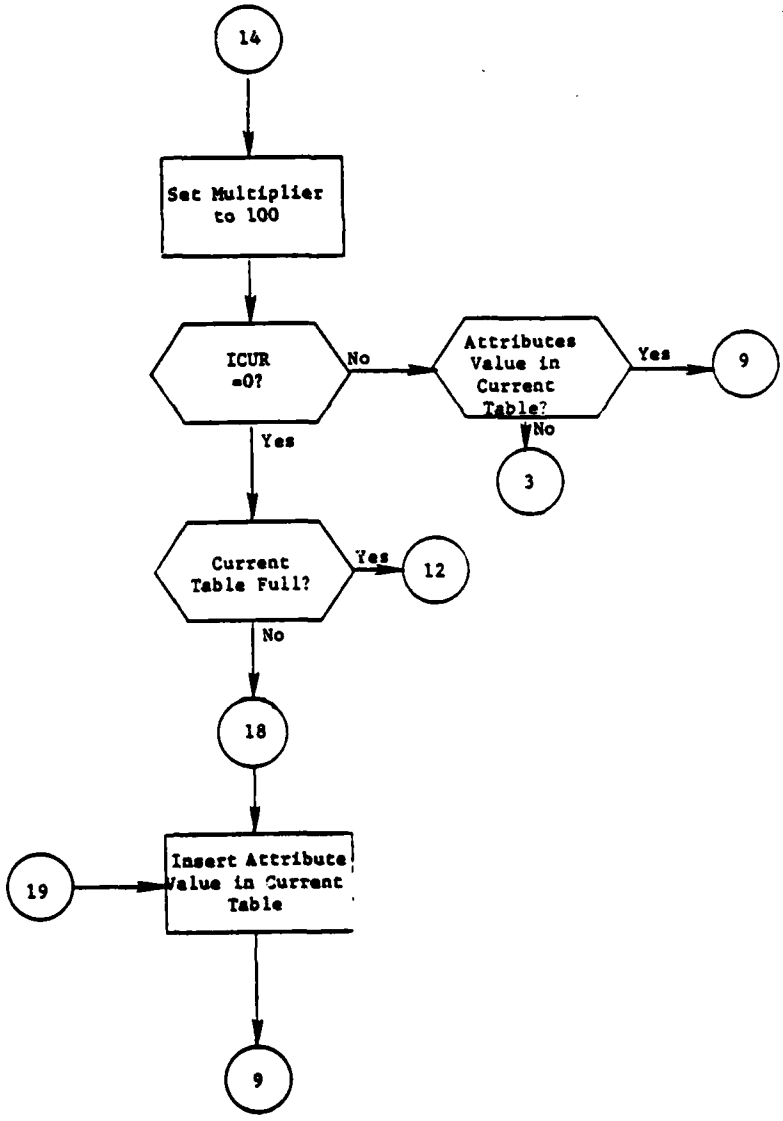


Figure 31. (Part 5 of 6)

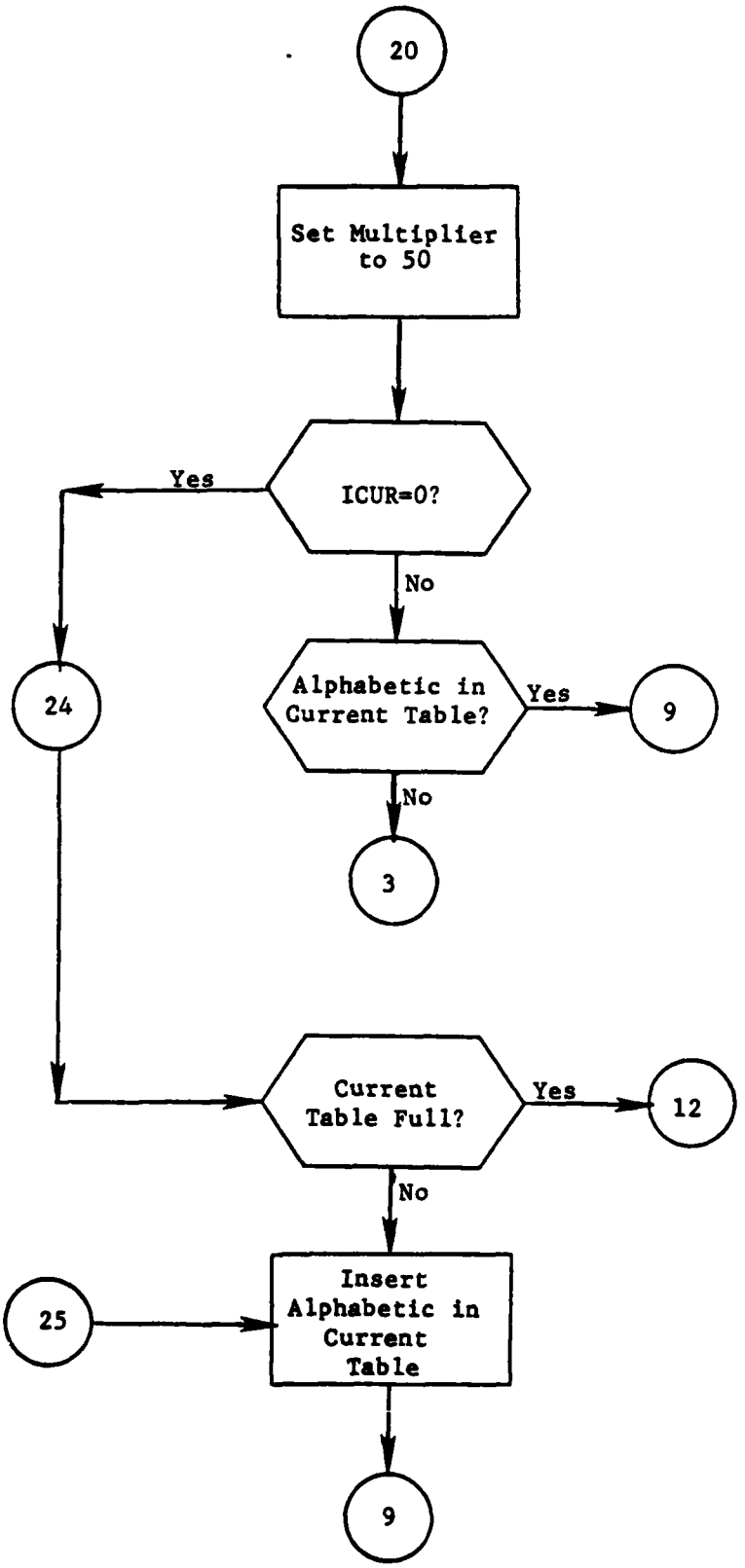


Figure 31. (Part 6 of 6)

3.9.4 Subroutine WEBSTR

PURPOSE: Looks up input strings in dictionary

ENTRY POINTS: WEBSTR

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C15, C30, STRING

SUBROUTINES CALLED: HDFND, NEXTTT, RETRV

CALLED BY: ERRFND, INICOP

Method:

First the tab character is created from the first two characters of ALPHA (from common block STRING). ISWT is set to assure a complete search of the TAB chain. The TAB chain is thus searched for the tab character. If the tab is not found, the process ends. If it is found, the WORD chain is searched for a match for ALPHA. If a match is found, TXPE and VALUE are set.

Subroutine WEBSTR is illustrated in figure 32.

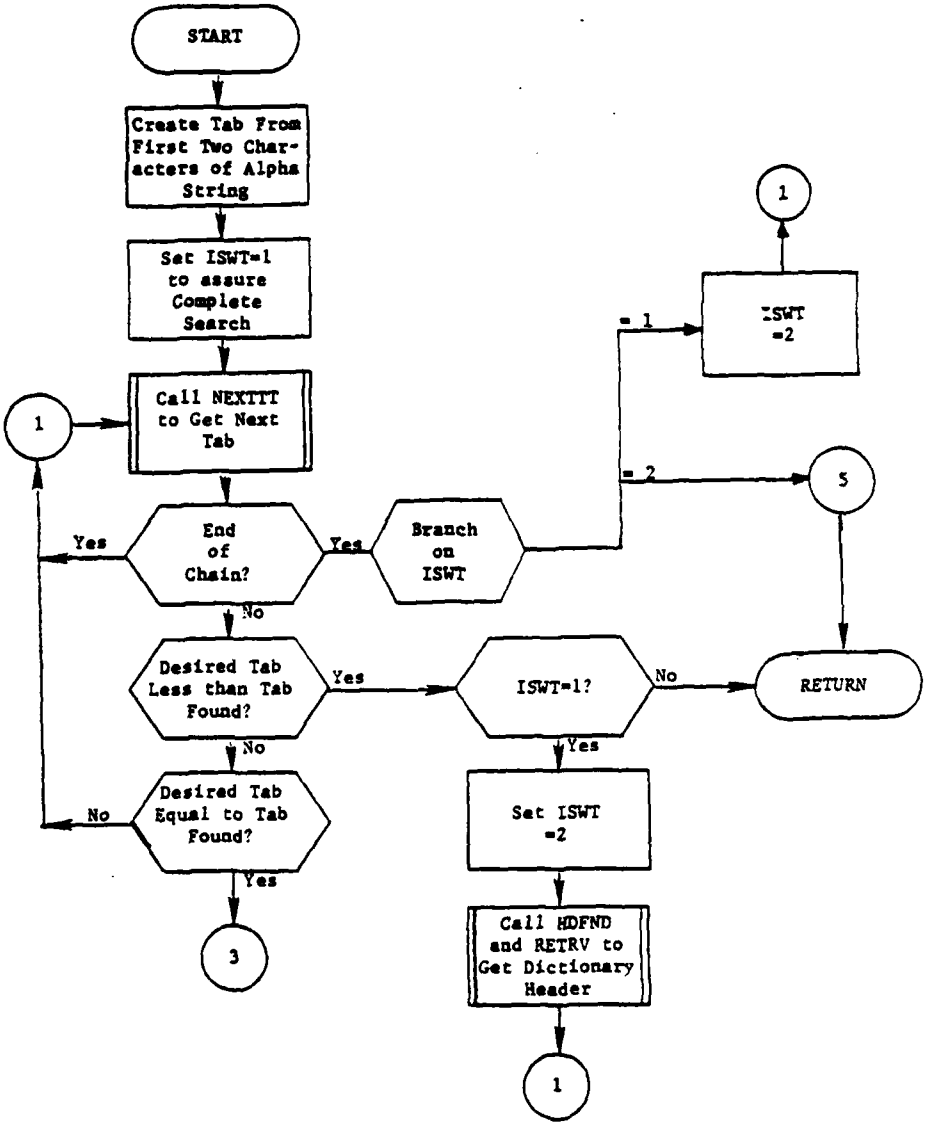


Figure 32. Subroutine WEBSTR (Part 1 of 2)

when a comma is encountered, the last left parenthesis of the proper level and the next right parenthesis of the same level have their symbols changed. Left parentheses symbols are changed so that bits 30-35=21, right parentheses symbols are changed so that bits 30-35=22.

Passes Two through Four are performed for each adverb in the order in which they were input. When all adverbs have been processed and the instruction table is complete, INSFLS is called to save the instruction table. DELTAB is called to delete ERRFND tables. The contents of the STRING common block are now restored and the subroutine exits.

Pass Two

The first step is to set up the pointer in the instruction prefix. If the adverb is null (JADEX=4) this is all that is done. A different process is used for adverbs with elemental phrases (JADEL=3). For these adverbs, each symbol of the clause in turn is converted to the appropriate "follower instruction."

If the adverb's phrases are relational phrases this pass is used to translate mathematical calculations and to replace any AND or OR operators which are, in reality, connectors for EQUALS and BETWEEN relations. In this process the branch RELOP is used to keep track of the part of the relational phrase expected next. Values for RELOP are:

RELOP = 1 - looking for an operator
RELOP = 2 - looking for a collector or object
RELOP = 3 - looking for object
RELOP = 4 - looking for continuation

When the beginning of the object is found, it is noted. While an object is being scanned the presence of any math operator is noted. When the end of the object is found, the processes branches depending upon whether any math operators were noted (MATH=true). If so, the code beginning at statement 55 (see figure 33) is used. If not, all parentheses in the object are removed.

The code at statement 55 is a process whereby each level of parentheses is resolved separately. The first step is to call PARLEV which flags each parenthesis with its level and notes the highest level. Then each level of parentheses is converted to instructions needed to calculate the value of each expression within the parentheses. The series of instructions is terminated by the instruction to store the calculated value in an internal variable. The parenthetical expression in the symbol table is replaced by a single non-zero symbol indicating the index number of the internal variable (bits 30-35=25, bits 0-29=index). As each level is evaluated more of the expression is removed until finally only one non-zero symbol for the internal variable containing the result remains.

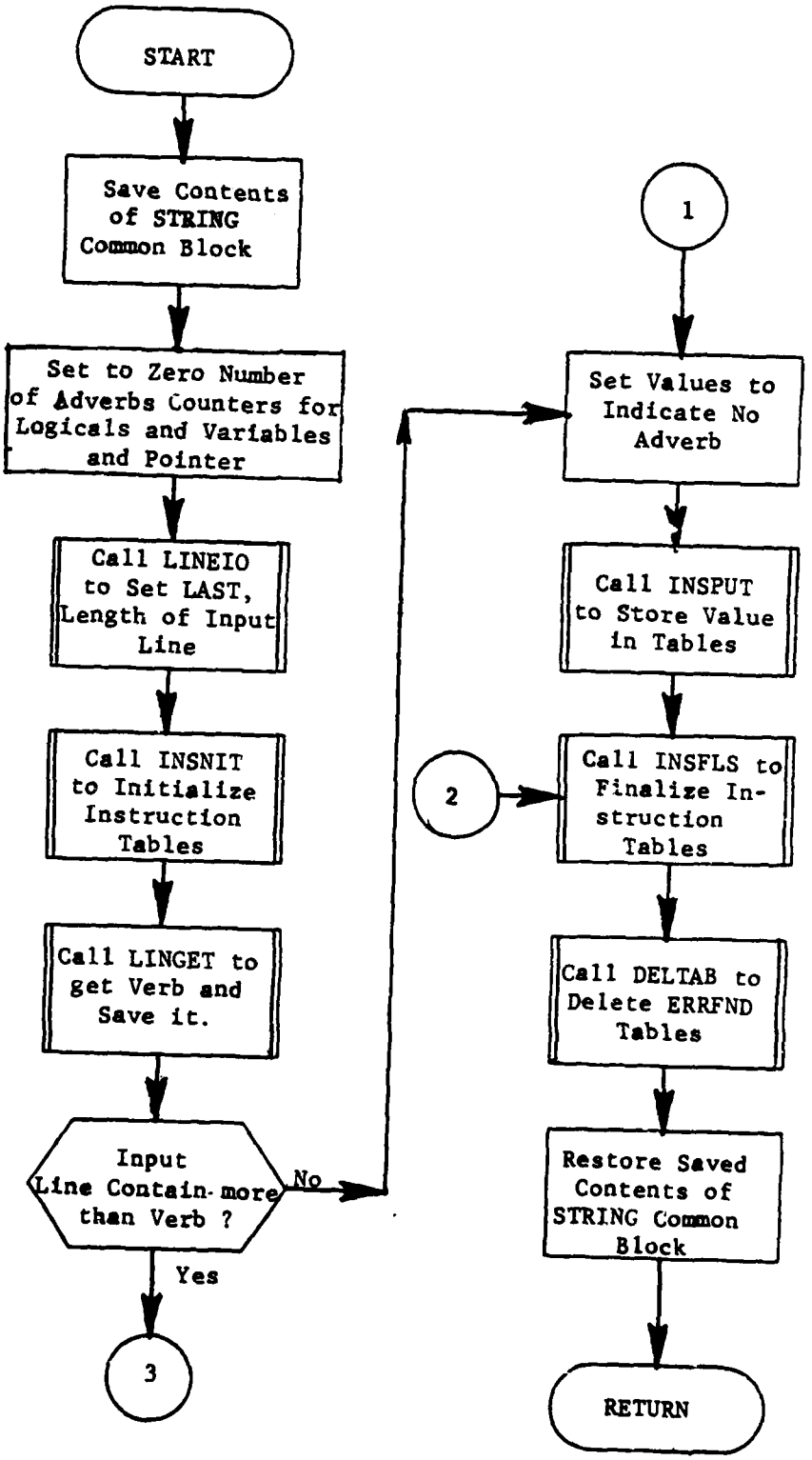


Figure 33. Subroutine INPTRN (Part 1 of 36)

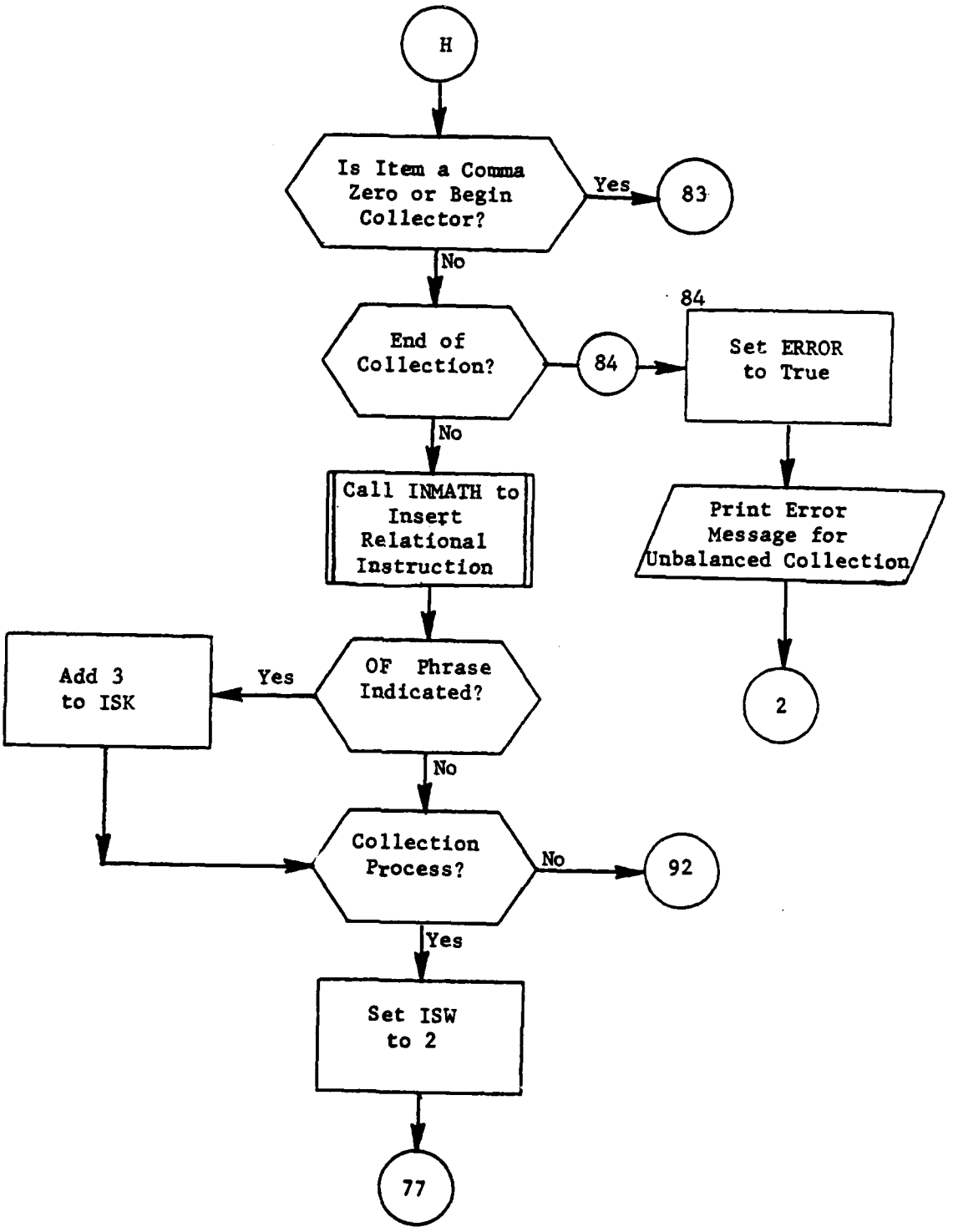


Figure 33. (Part 30 of 36)

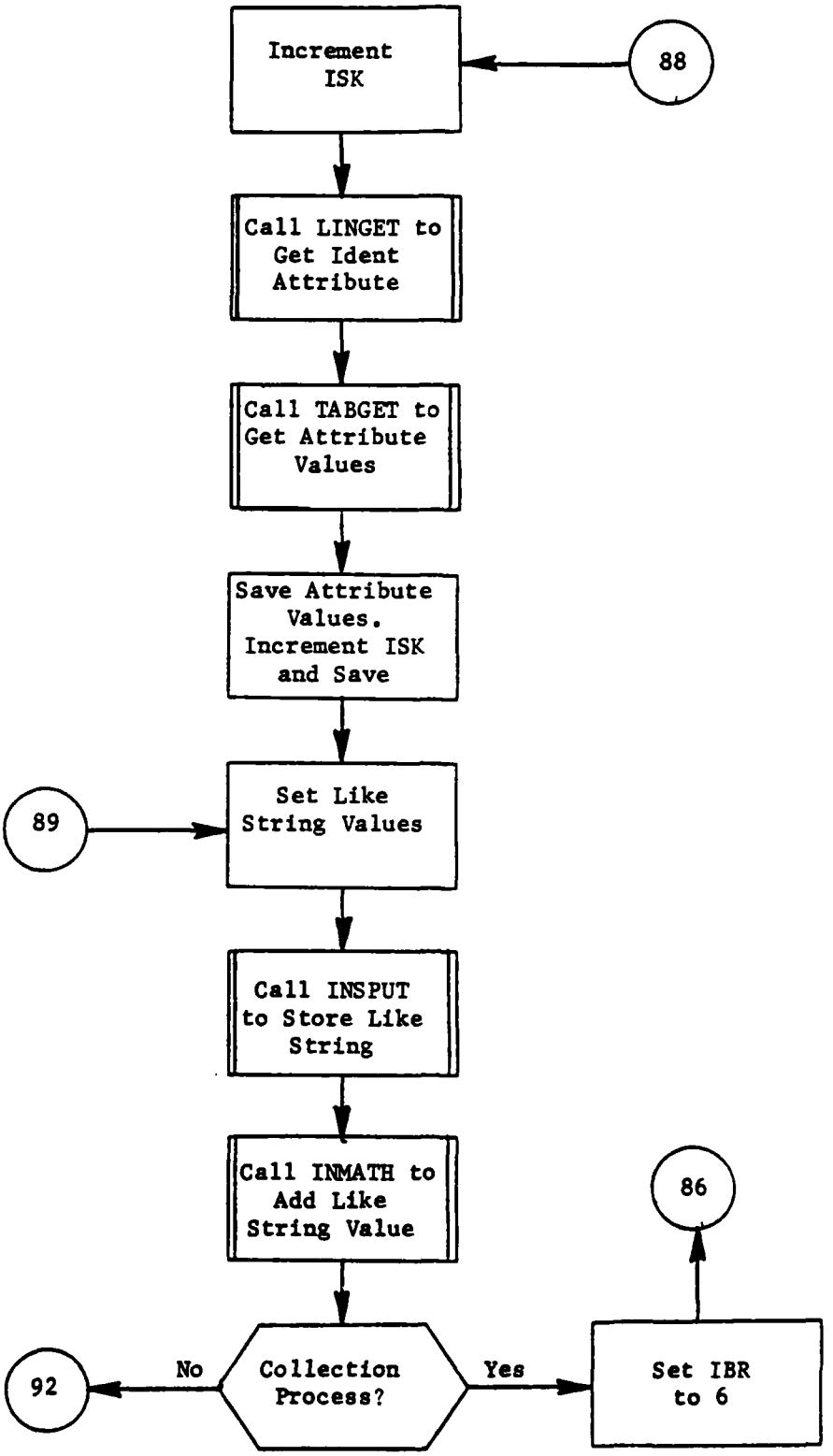


Figure 33. (Part 31 of 36)

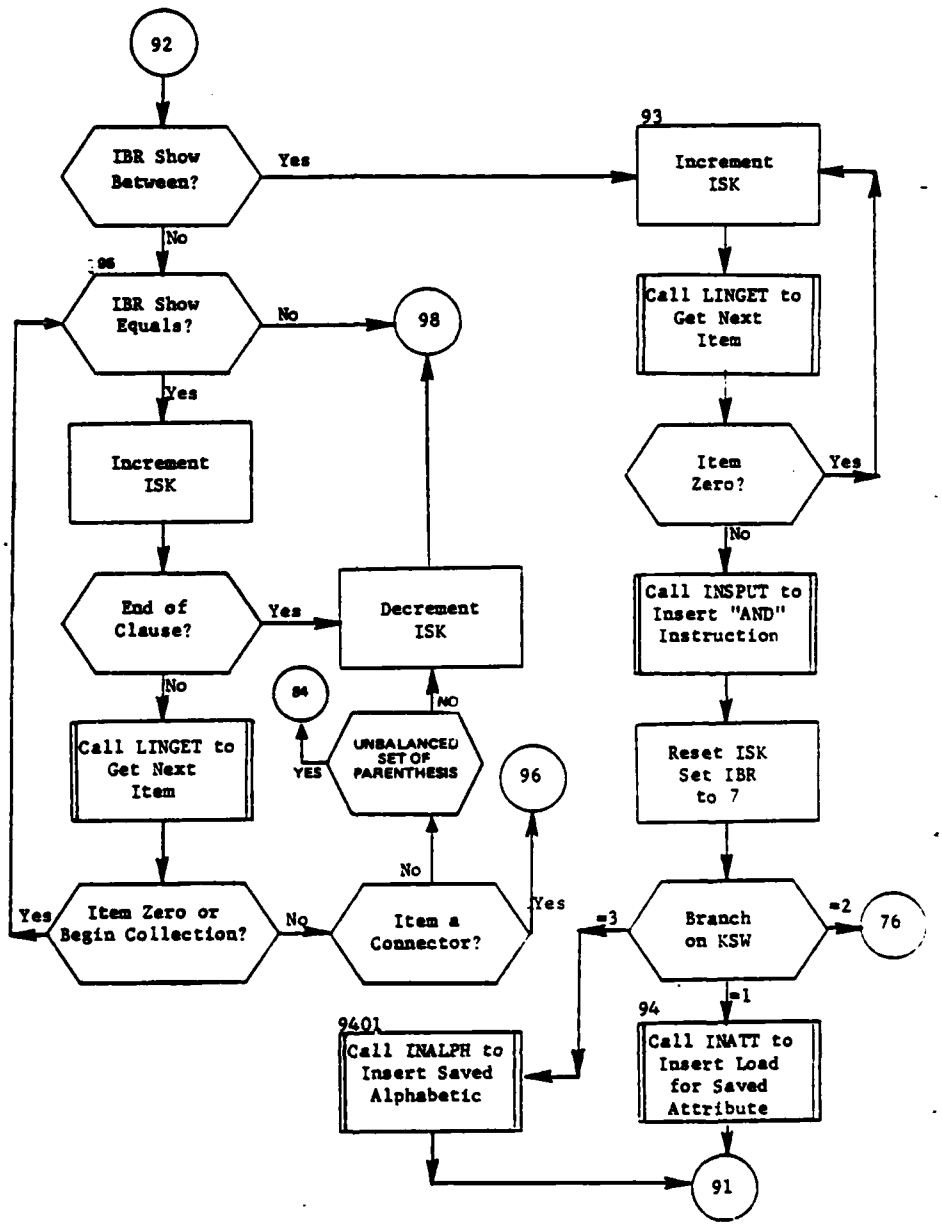


Figure 33. (Part 32 of 36)

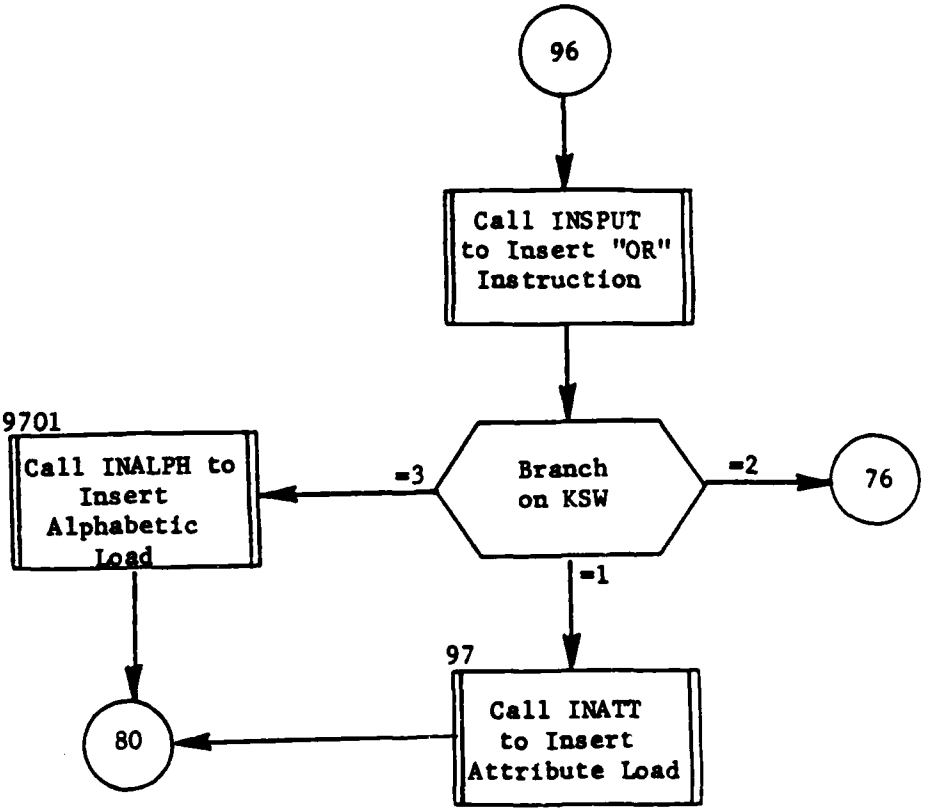


Figure 33 (Part 33 of 36)

3.10.5 Subroutine TABGET

PURPOSE: Obtain values from ERRFND tables

ENTRY POINTS: TABGET

FORMAL PARAMETERS: IXTYP: Type of table
INDX: Index for that table type

COMMON BLOCKS: C40, STRING, TABLZ

SUBROUTINES CALLED: RETRV

CALLED BY: INMATH, INPTRN

Method:

This subroutine obtains values from one of three ERRFND tables. The type of tables, indexed by IXTYP, is:

1 - Numeric constants
2 - Attributes
3 - Alphabetic constants

The process is to calculate which of the tables of the specified type is involved by dividing the input index (INDX) by the multiplier (100 for numeric or attribute, 50 for alphabetic). If this table number is greater than the number of "old tables" the value is obtained from the KTBVAL array. If the table is one of the "old tables" the old table in question is retrieved if it is not the current table and the value obtained from common block C40. The retrieved values are stored in the appropriate variable of the STRING common block.

Subroutine TABGET is illustrated in figure 38.

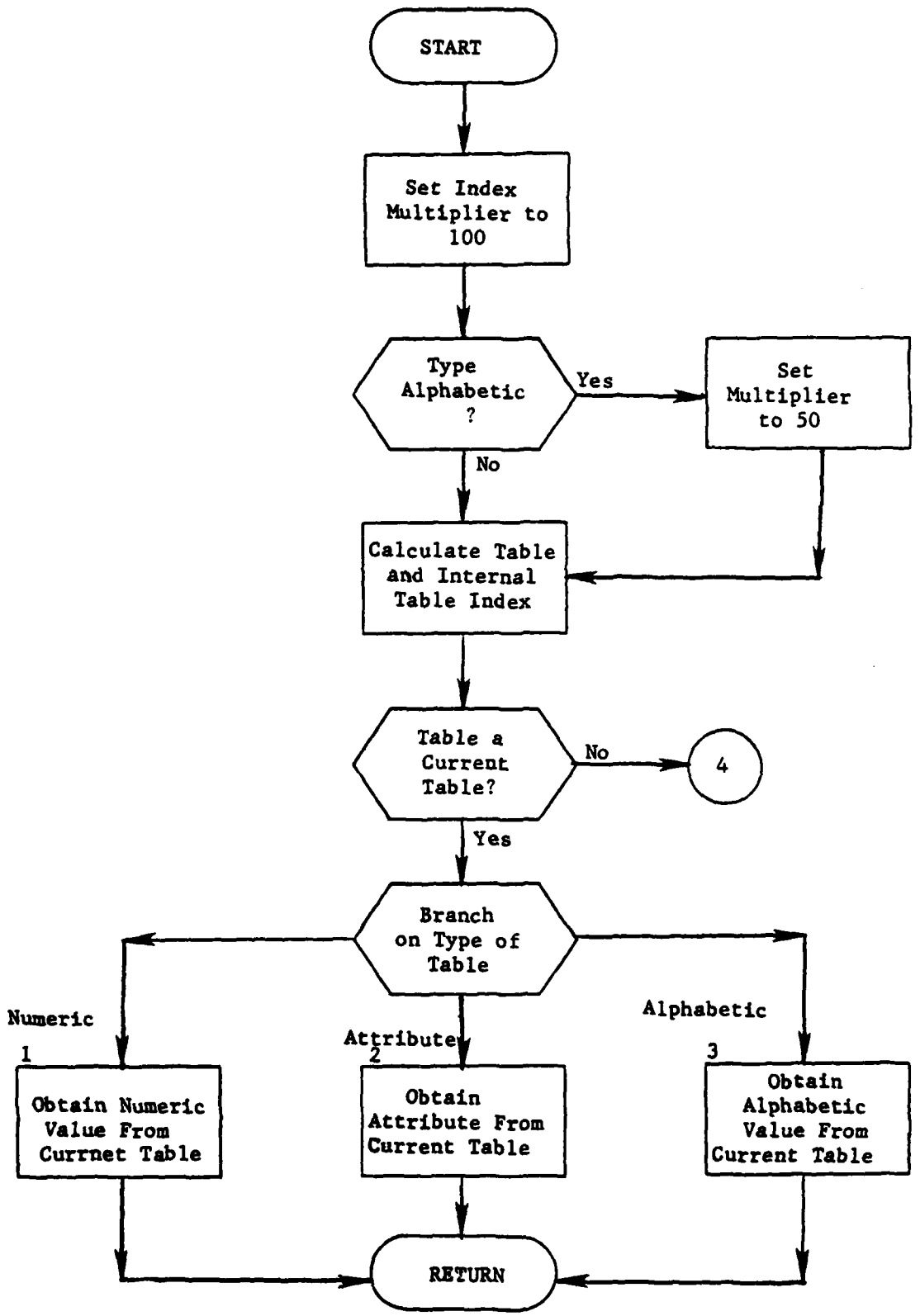


Figure 38. Subroutine TABGET (Part 1 of 2)

SECTION 4. DATA MODULE

4.1 Purpose

The DATA module is designed to allow the user to create those portions of the data base not created by other modules, add information to existing record types which were only partially initialized by other modules, make corrections and updates to existing record types and delete record types which are no longer desired.

4.2 Input

The card image inputs to DATA are the user command sentences which are completely generalized to the degree possible. As with all modules, the entire integrated data base may be queried by DATA. The only data base precondition of DATA execution is the initialization of the organizational data structure. Naturally, however, logical consideration of DATA execution is a must. For instance, no record can be changed which does not already exist; the CHANGE verb will never cause new records to be created. An attempt to create any record which already exists will be ignored. Equally, an attempt to delete non-existant record types may not be honored.

4.3 Output

The output of DATA implies the updating (or deletion) of the contents of record types within the integrated data base. For creation, new records are added to the data base under those master and chains as directed by the user card image input. Through query of the organizational structure, DATA may readily properly place user requests. A change request alters the contents of the previously constructed records. In some cases a match key may have been changed which causes the moving of a record type from its existing chain to another chain.

4.4 Concept of Operation

The DATA entry module first determines which verb initiated the call. In general thereafter, the input clauses are scanned to find any included attributes which are used to determine the record types effected. Finally, whatever action the verb calls for is carried out through subroutines CREAAT, CHANGE, and DELETE.

Subsections given below describe various philosophies used in the exercising of any of the DATA verbs.

4.4.1 Retrieval Schemes. When a module wishes to retrieve a specific subset of the data base it is best to do so in an orderly fashion. That is to say, for each level of the hierarchy to be retrieved, all

associated elements of the lower levels should be retrieved before the next example of that level is retrieved. A method for doing this is the retrieval scheme.

A retrieval scheme is an array which contains a series of instruction like word groups. By following the instruction pattern contained in this array a portion of the data base will be retrieved, one unique logical set of record types at a time, until all desired logical sets have been retrieved. Each instruction consists of an introductory word which contains an identifying number, followed by one to three words which make up the remainder of the instruction. There are four instruction types: Get Header, Chain Next, Chain Master, and Return.

4.4.1.1 The Get Header Instruction. This instruction always occurs first in the scheme. It is the instruction that tells the executing module to look for the next data header. This instruction contains a maximum of four words depending upon the code in the second word. The first word contains a 1. The second word informs the executing module whether the attribute CLASS or SIDE (or both) are to be checked. It has the following meanings:

1 - do not check CLASS or SIDE
2 - check CLASS only. Value for CLASS appears in word three
3 - check SIDE only. Value for SIDE appears in word three
4 - check both CLASS and SIDE. Value for CLASS appears in word three; value for SIDE appears in word four

4.4.1.2 The Chain Next Instruction. This instruction directs the executing module to retrieve the next record on a chain and informs it as to which instruction is to be executed if the master of the chain is retrieved. The instruction always has four words. The first word contains a 2. The second word contains the name of the chain. The third word contains the record type number of the chain's master. The fourth word contains a pointer (index number) to the instruction to be executed if the master of the chain is retrieved. In most cases, the pointer will be to the previous Chain Next Instruction or the Get Header Instruction.

4.4.1.3 The Chain Master Instruction. This instruction directs the executing module to retrieve the master record of a chain. The instruction always has two words. The first word contains a 3. The second word contains the name of the chain.

4.4.1.4 The Return Instruction. This instruction always appears last in a scheme. It informs the executing module that a logical set of records has been retrieved. It also informs the module of the instruction to execute next to retrieve the next logical set of records. The instruction contains two words. The first word contains a 4. The second word contains a pointer to a previous instruction.

4.4.1.5 Retrieval Supporting Subroutines. Retrieval schemes are built by utility subroutines SETSCH from a continuous set of record type numbers. By continuous is meant that every record type in the set is either the master or detail of a chain or series of chains by which it is linked to every other record type in the set. The utility subroutine LINKUP is designed to assist in this process. To build a retrieval scheme LINKUP and SETSCH also require that one of the record types in the set be identified as the 'primary header'.

4.4.2 Primary Header. The QUICK integrated data base is designed with the idea that all data will be retrieved through the use of data entry point record types known as headers (see section 2). The set of record types from which a retrieval scheme may be built, might conceivably contain any number of such headers. Therefore, one of the headers must be designated as the data entry point and is referred to as the 'primary header.' The DATA module uses several methods to determine the primary header. One is to use the desired value for the attribute CLASS. Since this attribute uniquely defines a header, DATA assumes this is the primary header. Another method is to choose some lower level record type in the hierarchy. This record may be determined by record type number since as the record type numbers increase the hierarchial level tends to decrease or, in the case of CREAAT, by the record type with the greatest number of involved attributes. In either case utility subroutine PRIMHD is called. This subroutine traces chains of which the record type is a detail up to a header. This header is then used as the primary header.

4.4.3 Determining a Record Type Set from a List of Attributes. The set of record type numbers is built by using the list of attributes which have been included in the input clauses. Attributes are of three types in terms of the record types that include them as described below:

Single - This type of attribute is included on only one record type.

Control - This type of attribute is included on several record types but one of the record types is uniquely identified by the attribute. The record type so identified is referred to as "controlled."

Multiple - This type of attribute is included on several record types but is not a unique identifier for any of them.

The general method of determining a record set from the list of attributes is to make two record type lists. The first or main list includes all those record types which contain single attributes in the attribute list plus the controlled record types of any control attributes in the attribute list. The second or multiple list includes all record types which include any of the multiple attributes in the attribute list. The primary header is now found either from a value for CLASS or from

one of the record types in the main list. The primary header is then used in a 'chain-down' process. This type of process consists of checking every record type which is a detail of chains of which the primary header is the master. Then using these detail records as masters of chains and so forth until the lowest levels of the hierarchy headed by the primary headers have been reached. Throughout the process, the multiple list is examined and anytime a record type in the hierarchy appears in the multiple list it is added to the main list. The main list is then used as the set of record types from which a retrieval scheme is built.

4.4.4 Data Queues. In the process of the CREAAT or CHANGE subroutines often the subroutine must build one or more data queues. This occurs when an attribute or collection of attributes are given more than one value or collection of values. The queue is an ordered list of the values given to the attribute or collection and is maintained by the subroutine VALPUT. Each value assigned to a particular attribute can be identified by its position in the queue. Each value assigned to the member of a collection can be identified by its position within the collection plus its position in the queue minus one times the length of the collection.

The CREAAT subroutine uses the set of queues it has created to determine how many times it must go through the process of creating record types. For each combination of values in queues, the process must be followed. For example, if there is one queue for a single attribute with three values and another queue for a collection of four attributes with five sets of values (a total of 20 values), the record creation process would be performed 15 times. CREAAT uses a set of queue counters which are all originally set to 1 indicating the first value or set of values in each queue. After every execution of the record creation process, the last queue counter is incremented unless it is at maximum. In this case it is reset to one and the next to last counter incremented. If the next to last is at maximum it is reset to one and the prior counter incremented and so on. In this way all combinations are used in the record creation process.

The CHANGE subroutine expects that if a data queue is formed from the SETTING clause one of the same length will be formed from the WHERE clause. Then when a record is retrieved which satisfies the WHERE clause the data queue for the WHERE clause is searched until the value (or collection of values) which caused the match is found. The value (or collection of values) which occupies the same ordinal position in the data queue for the SETTING clause is used in the record change process.

4.5 Identification of Subroutine Functions

4.5.1 CREAAT. This subroutine (figure 39) performs the functions of creating new records. After checking for the existance of SAME and SUPRESSING adverbs, CREAAT scans the SETTING clause (any number of SETTING clauses may be input, each is processed similarly). First the attributes are collected, calculations flaged and OF phrases resolved. Next, attribute values are saved and data queues built for attributes provided with multiple values. Next a retrieval scheme is built by identifying the record types defined by the attributes. Input values are checked against QUICKs directory. Attributes on records to be created but not given values are assigned values either according to the defaults in the directory or from the record type(s) identified by the SAME clause. Finally, records are created of every type identified except in cases where the record to be created would duplicate an existing record.

4.5.2 CHANGE. This subroutine (figure 40) performs the function of changing existing records. First the SETTING clause is scanned and attributes and values are collected. If a data queue exists it is set up. Next the WHERE clause is scanned and its attributes are collected. The two clause's lists are combined with any WHERE clause queue paired with its SETTING clause counterpart. The list of attributes is used to build a retrieval scheme. The retrieval scheme is now executed and any record type combination retrieved which satisfies the WHERE clause is modified according to the SETTING clause.

4.5.3 DELETE. This subroutine performs the function of deleting unwanted records. First the WHERE clause is scanned for attributes. Next a retrieval scheme is built using the attributes found. From this scheme, the record lowest in the hierarchy is determined. Finally, the scheme is executed and for every record type combination retrieved which satisfies the WHERE clause, the lowest record type in the hierarchy is deleted.

4.6 Common Blocks

The common blocks internal to the DATA module appear in table 15.

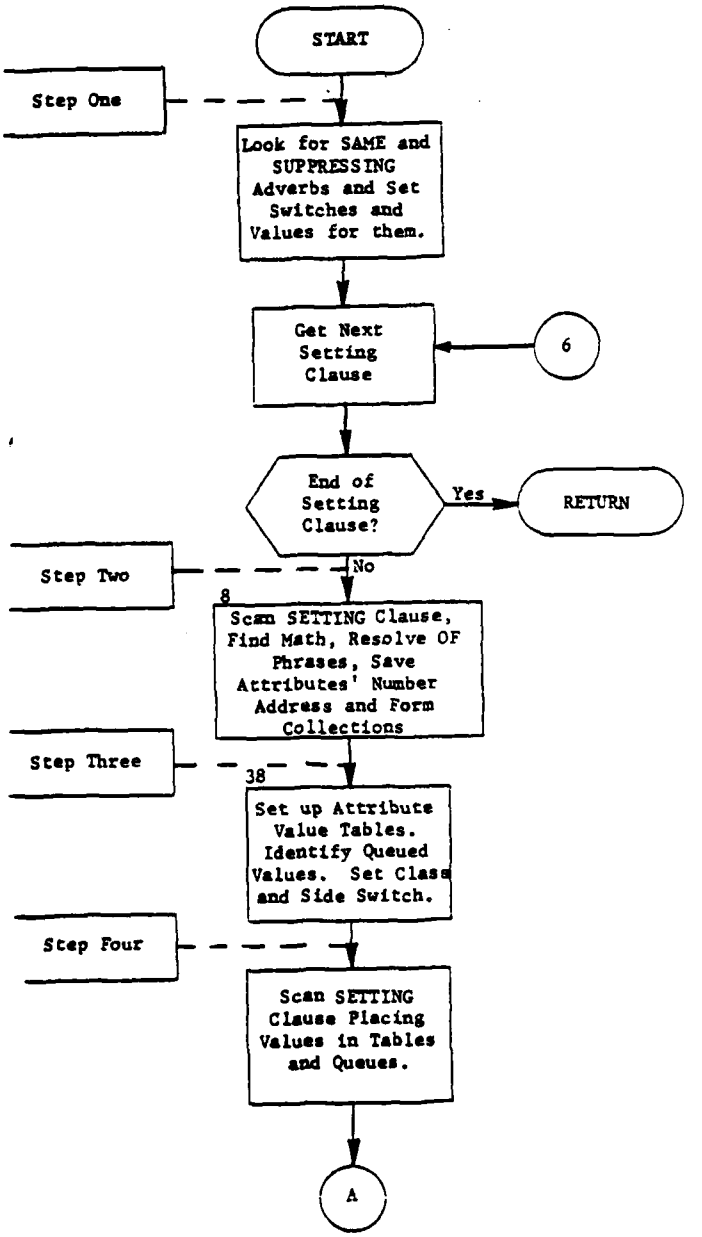


Figure 39. Subroutine CREAAT: Macro Flow (Part 1 of 3)

Table 15. DATA Module Internal Common Blocks

BLOCK	ARRAY OR VARIABLE	DESCRIPTION
ORDER	SCHORD(100)	Record type numbers of record types involved in retrieval scheme in retrieval order
	SORDNM(100)	Record type names in retrieval scheme order
	LENSCH	Length, in words, of retrieval scheme
PRINSP	PRINON	Switch to control optional prints True - produce prints False - do not produce prints
SCHEME	POINT	Pointers to current instruction of retrieval scheme
	SCHEME(200)	Retrieval scheme
SCRTCH	LIST(300)	Storage space used as work area by several subroutines

4.7 Subroutine ENTMOD

PURPOSE: Entry subroutine for DATA module

ENTRY POINTS: ENTMOD (first subroutine called when overlay DATA is executed)

FORMAL PARAMETERS: None

COMMON BLOCKS: PRINSP, QC

SUBROUTINES CALLED: CHANGE, CREAAT, DELETE, INSGET

CALLED BY: MODGET

Method:

This subroutine performs the function of selecting the desired DATA module overlay. INSGET is used to obtain the value of the verb which caused the call. The adverbs are scanned for the ONPRINTS adverb. If it is found the PRINON variable in the PRINSP common block is set to true. Then the requested overlay link is read in and the verb executed.

Subroutine ENTMOD (DATA) is illustrated in figure 41.

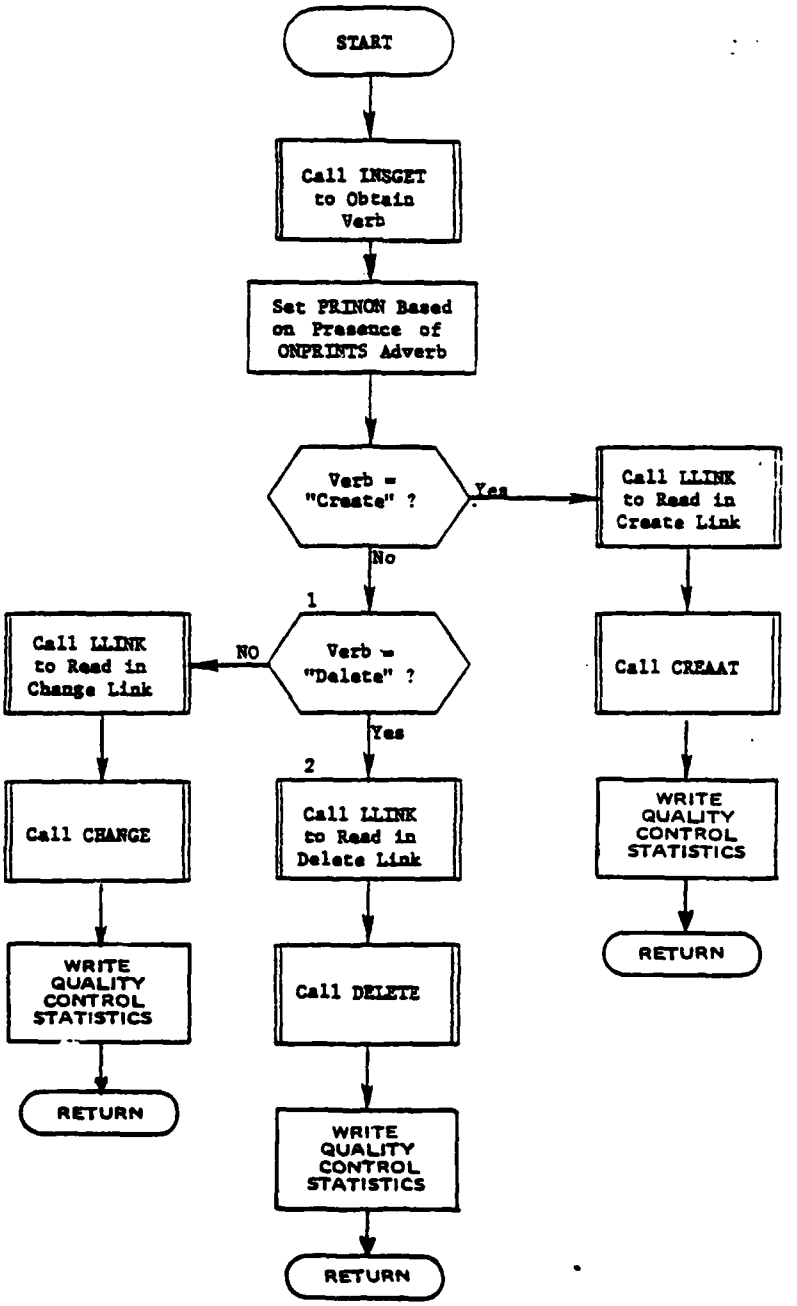


Figure 41. Subroutine ENTMOD (DATA)

4.7.1 Subroutine VALPUT*

PURPOSE: Store values for data queues

ENTRY POINTS: VALDEL, VALGET, VALPUT

FORMAL PARAMETERS: VALUE: Value to be stored or value retrieved
COUNT: Position of value in queue
COLECT: Identifying number of queue

COMMON BLOCKS: C40

SUBROUTINES CALLED: DLETE, RETRV, STORE

CALLED BY: CHANGE, CREAAT

Method:

Entry Point VALPUT

The input VALUE is entered in the current entry space. The current pointer for the COLECT queue is also stored with VALUE and the queue count incremented. The index of VALUE is saved as the new current pointer for the COLECT queue. If the buffer is full, a new TABLEZ is created and the pointer reset to 1.

Entry Point VALGET

The queue count and COUNT are used to determine the number of entries needed for search since the values are in the queue in reverse order. The number of entries is then retrieved starting with the current pointer and proceeding using the pointer in the entry until the desired entry is retrieved. Its value is set in VALUE.

Entry Point VALDEL

Any TABLEZs created are retrieved and deleted. Also, the queue counts and pointers are set to zero.

Subroutine VALPUT is illustrated in figure 42.

*This subroutine appears in overlays DATACH and DATACR.

4.8 Subroutine CHANGE*

PURPOSE: To change existing records

ENTRY POINTS: CHANGE

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C20, C30, ERRCOM, OOPS, ORDER, PRINSP, QC, SCHEME

SUBROUTINES CALLED: DESSCH, GETNXT, HDFND, HEAD, INSGET, LINKUP, MODFY, NEXTTT, NXTDES, OFVAL, PRIMHD, SETSCH, UNCODE, VALDEL, VALFND, VALGET, VALPUT, XLL, XMATH, XWHERE

CALLED BY: ENTMOD (DATA)

Method:

The CHANGE verb process may be broken into eight steps which are carried out in sequence (see figure 40).

Step One

The SETTING clause is scanned. In the process any OF phrases and LIKE strings are immediately resolved using VALFND and their values stored via OFVAL. Also the extent of any mathematical calculations is noted as the limits for execution for subroutine XMATH. The main thrust of the step is to list any attributes (ATNUMB) and save the values assigned to them. If an extended equals phrase is included, a queue of the values is built and the involved attribute(s) are flagged (ATTYPE=2) (see figure 43).

Step One and a Half

The SETTING clause is scanned. In the process, any attribute which is simply set to a value is looked up in the directory and the value checked. List alphabetics have their list of valid values compared to their input values. Numeric attributes have their values compared to the limits given in the directory. LAT and LONG are compared to hard-coded limits. Any errors are reported but processing continues.

Step Two

The WHERE clause is scanned. In the process any OF phrases and LIKE strings are immediately resolved using VALFND and their values stored via OFVAL. Any attributes are saved in a list (WHATNB). If an extended equals phrase is included, its involved attributes are flagged (WHATYP=4)

*The main routine of overlay DATACH

and its values stored in a queue via VALPUT. The CLASS and SIDE attributes are particularly noted and their values saved to assist in the retrieval scheme construction process. If the DESIG attribute is the only attribute included this fact is noted as the retrieval process differs greatly in this event (see figure 44).

THIS PAGE INTENTIONALLY LEFT BLANK

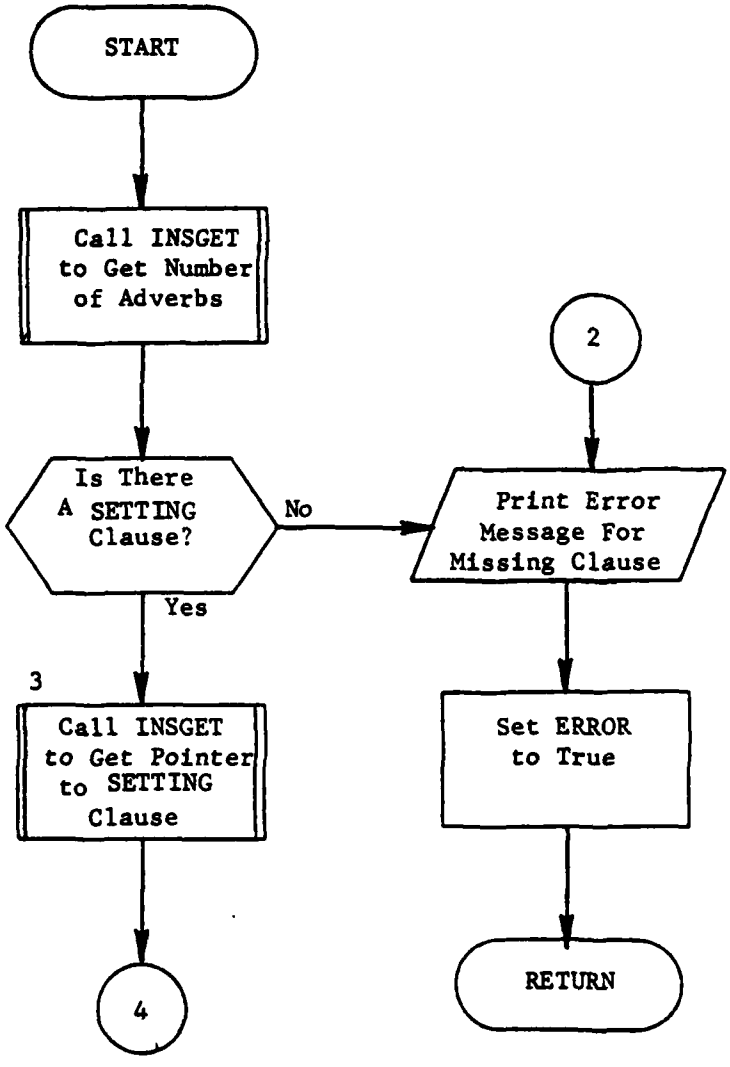


Figure 43. Subroutine CHANGE: Step One (Part 1 of 10)

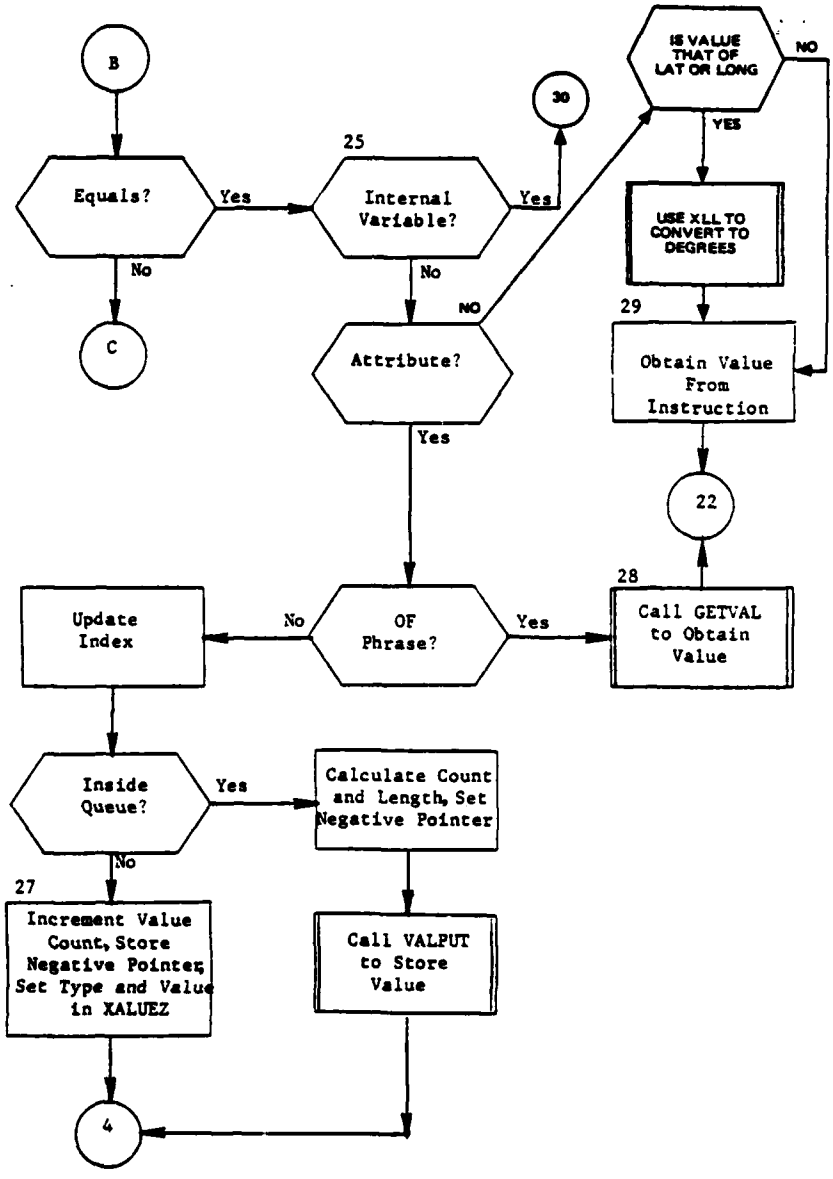


Figure 43. (Part 4 of 10)

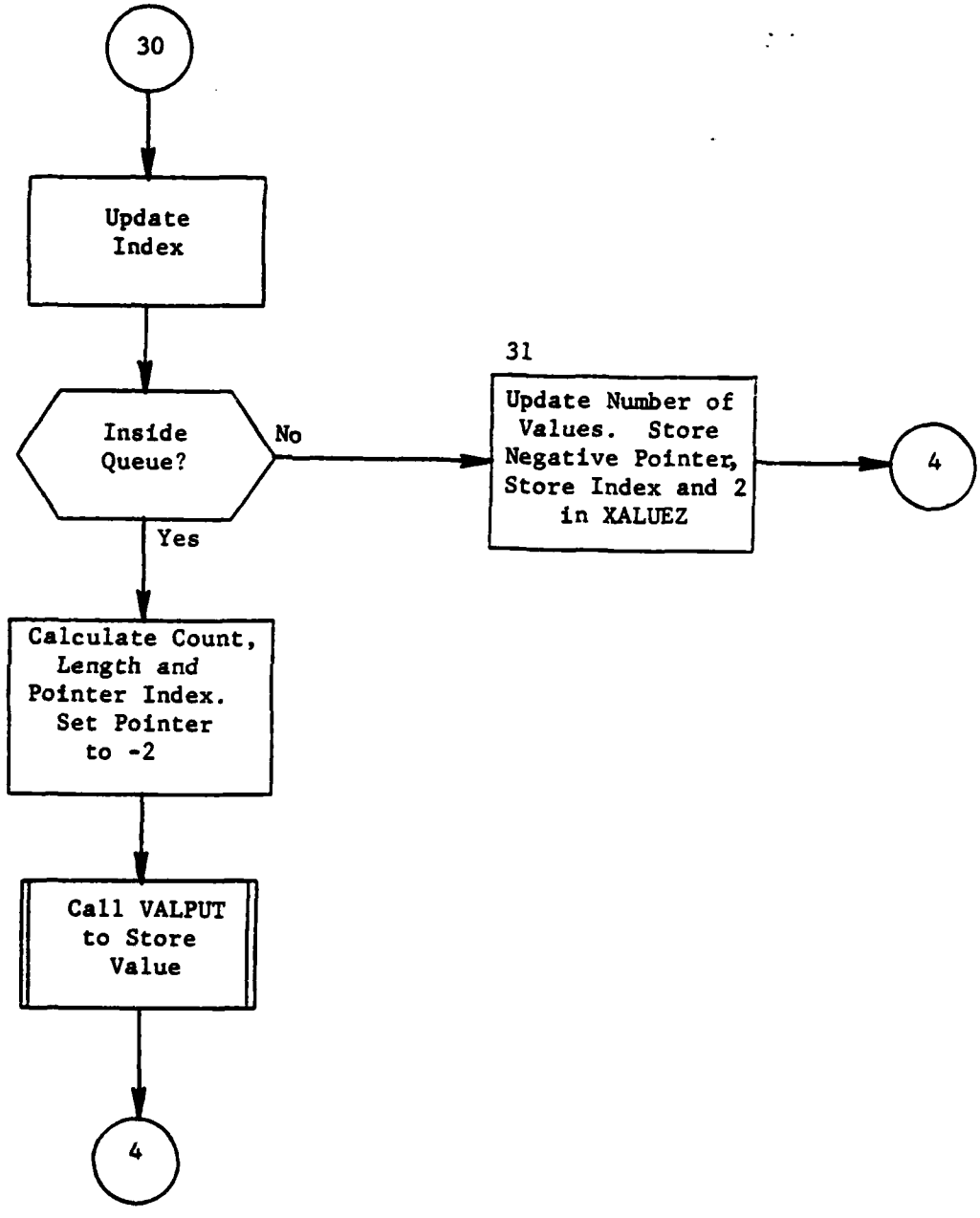


Figure 43. (Part 5 of 10)

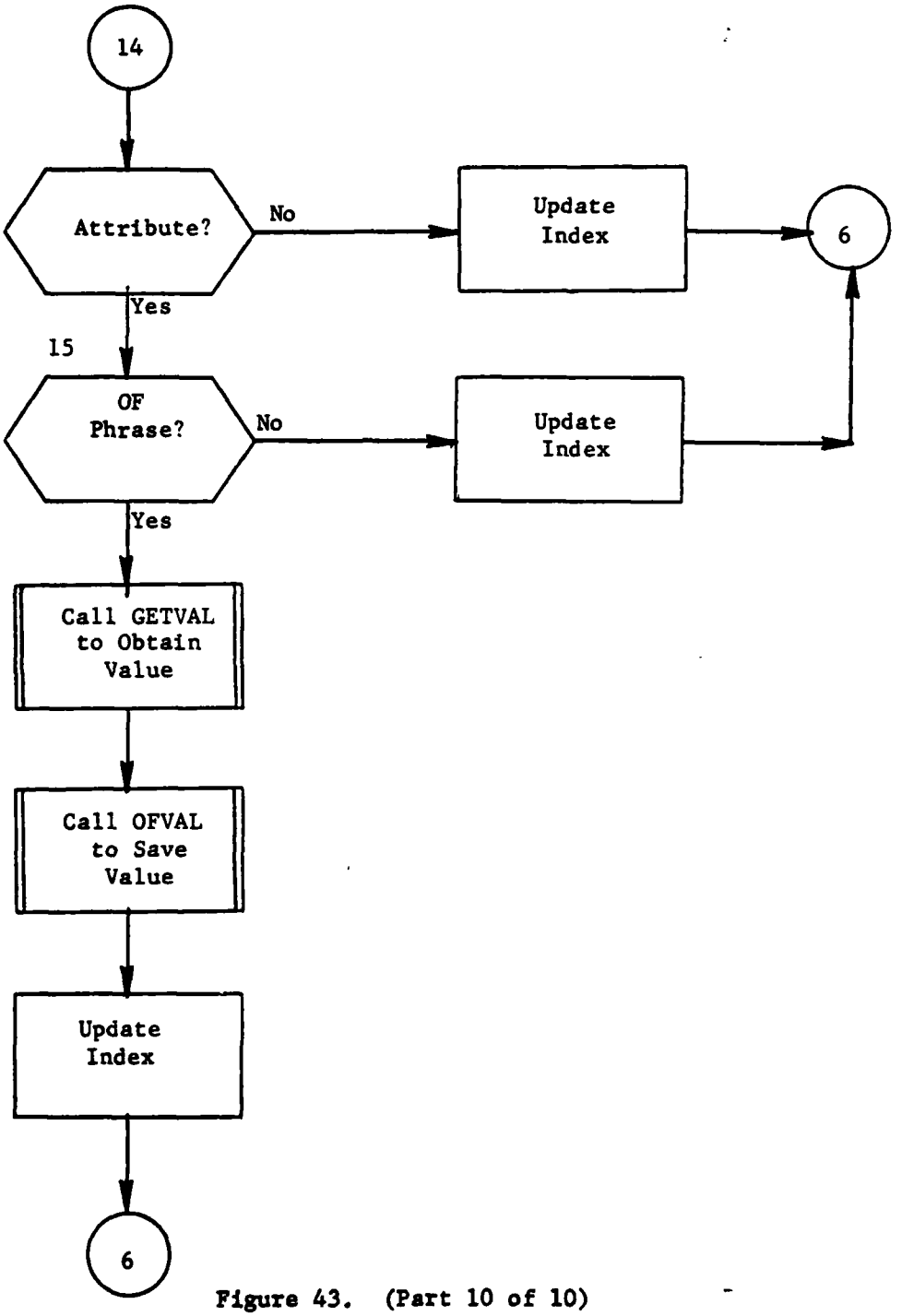


Figure 43. (Part 10 of 10)

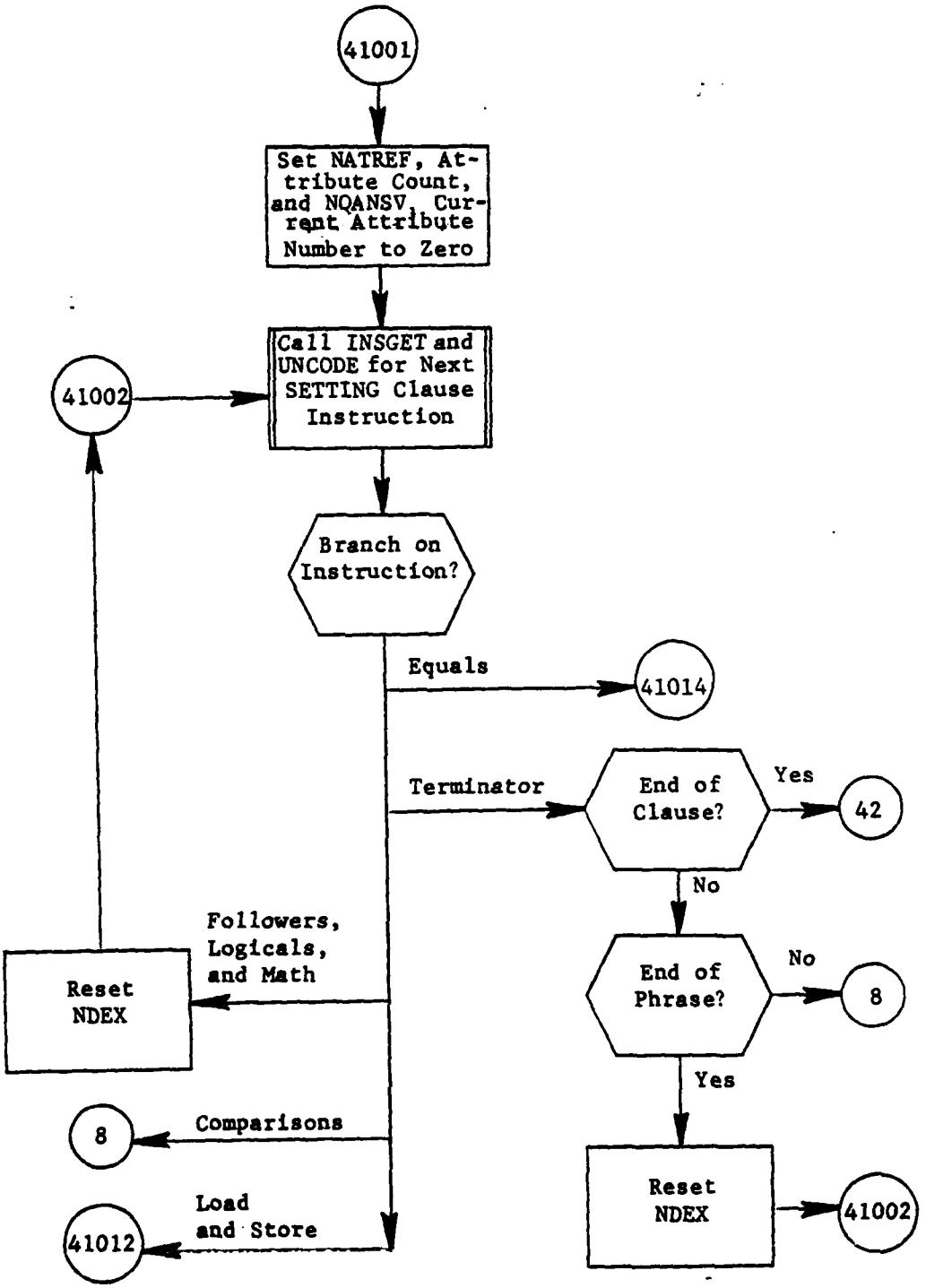


Figure 43.1. Subroutine CHANGE: Step One and a Half (Part 1 of 7)

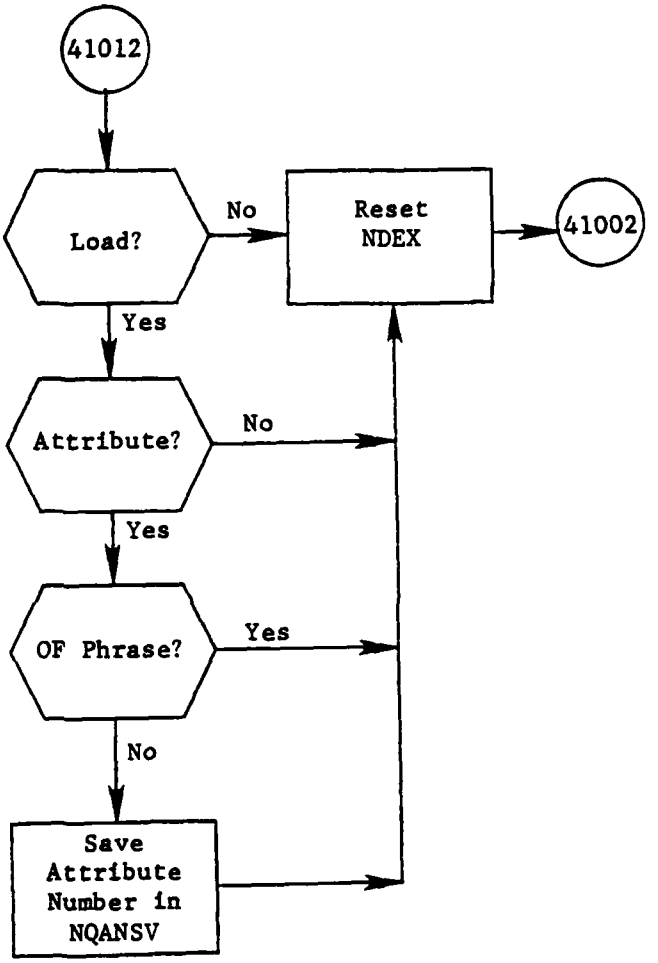


Figure 43.1 (Part 2 of 7)

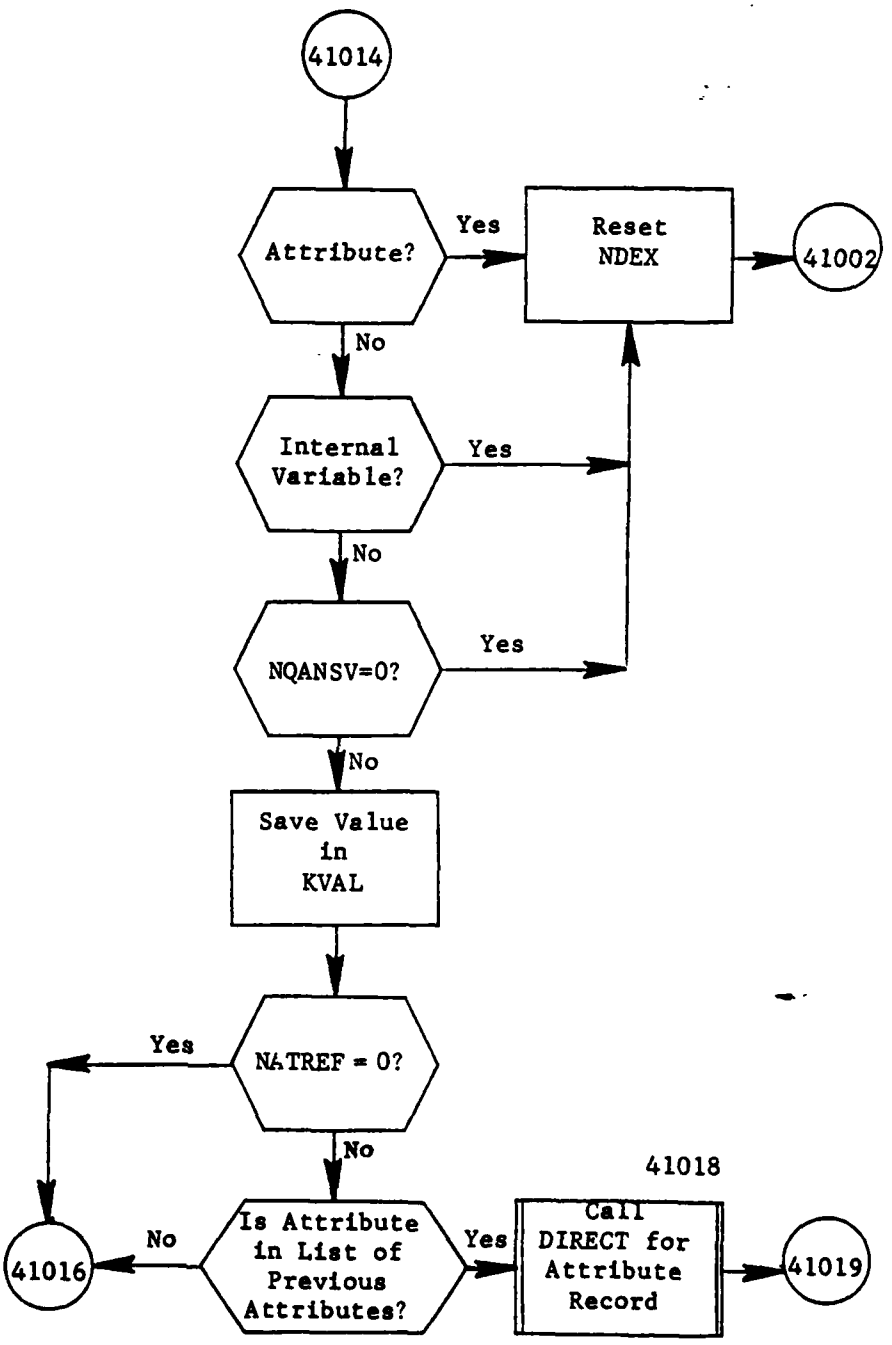


Figure 43.1. (Part 3 of 7)

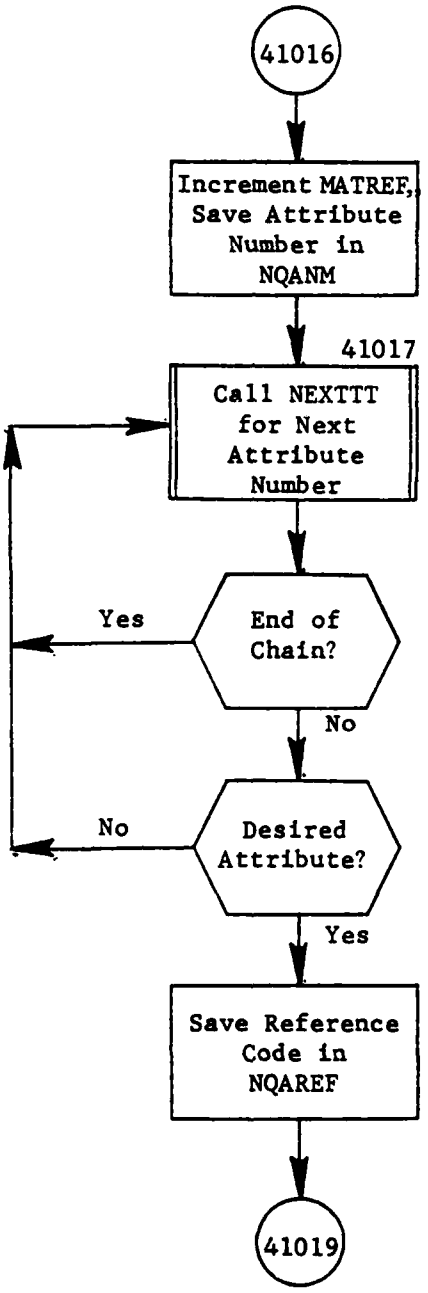


Figure 43.1. (Part 4 of 7)

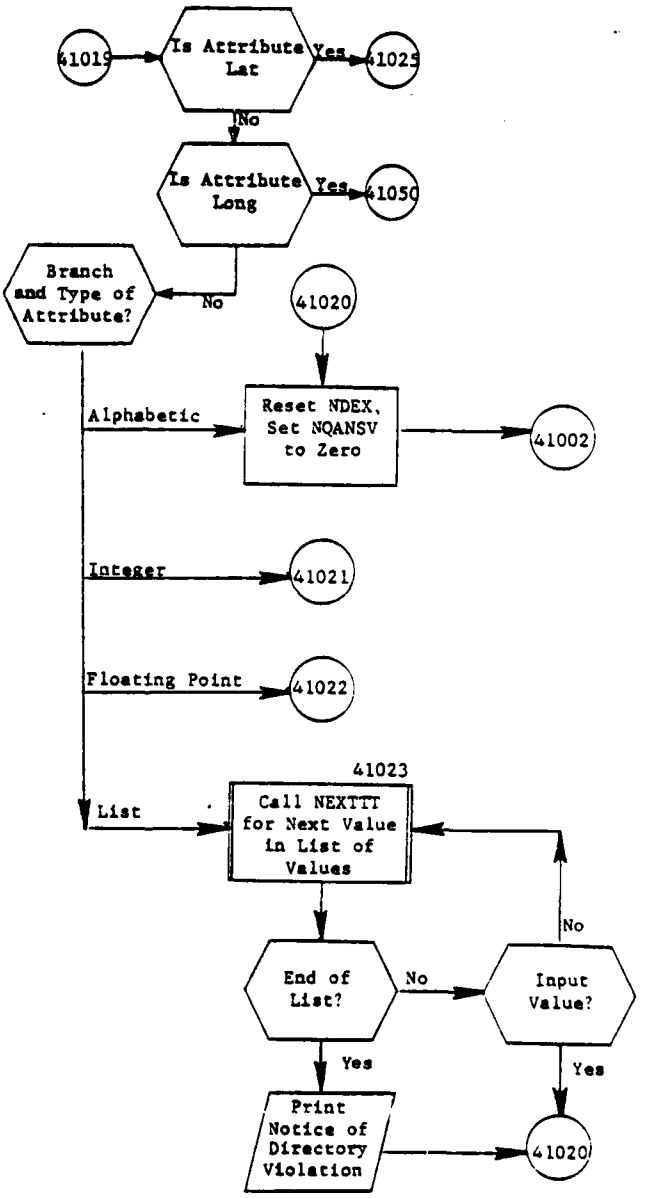
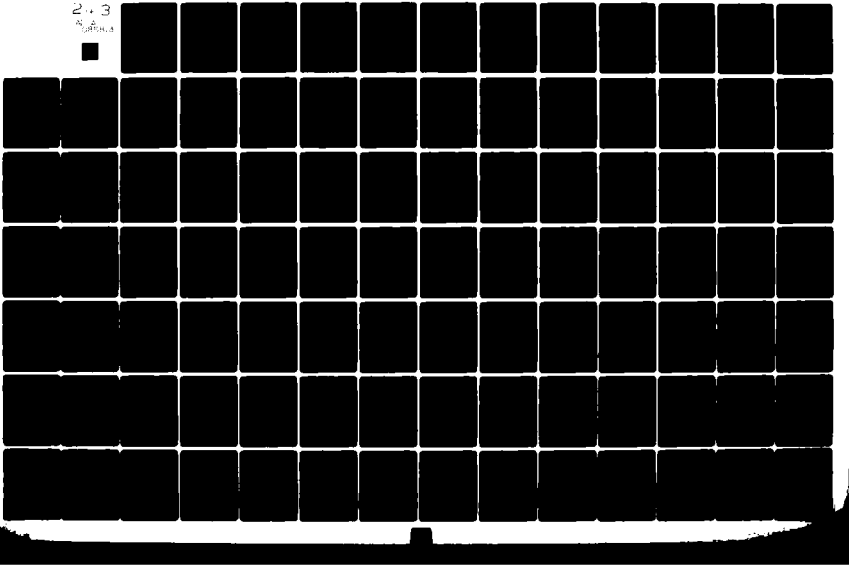
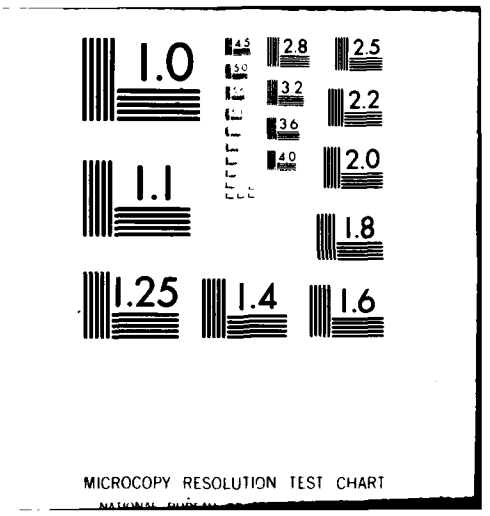


Figure 43.1. (Part 5 of 7)

AD-A085 813 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON DC F/G 15/7
THE CCTC QUICK - REACTING GENERAL WAR GAMING SYSTEM (QUICK) PRO--ETC(U)
MAY 80
UNCLASSIFIED CCTC-CSM-MM-9-77-V1-CHG-3 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART

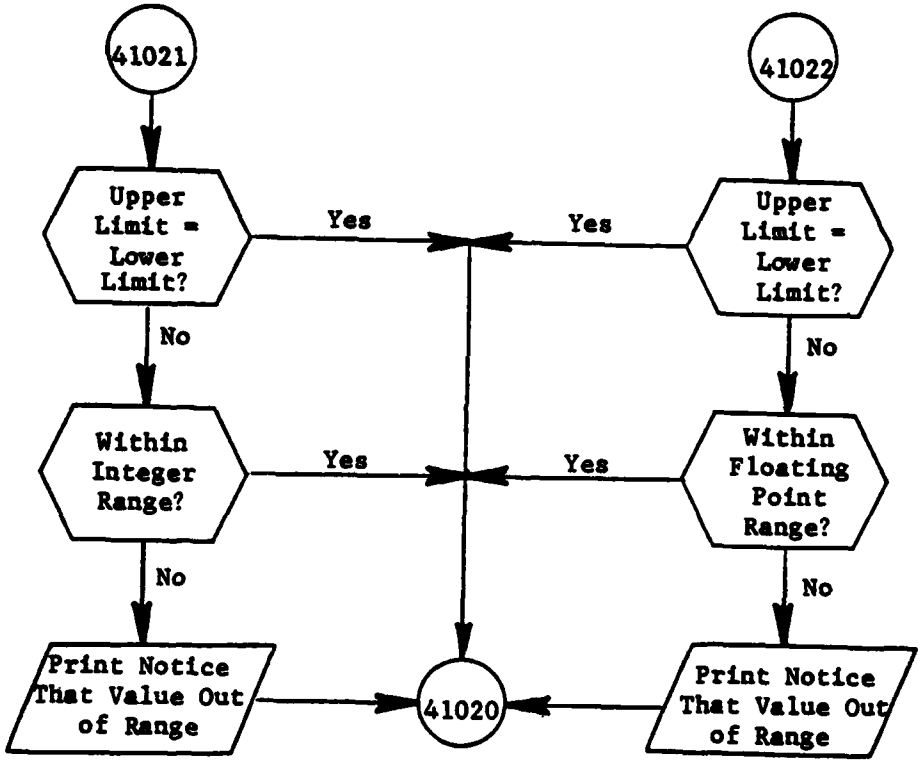


Figure 43.1. (Part 6 of 7)

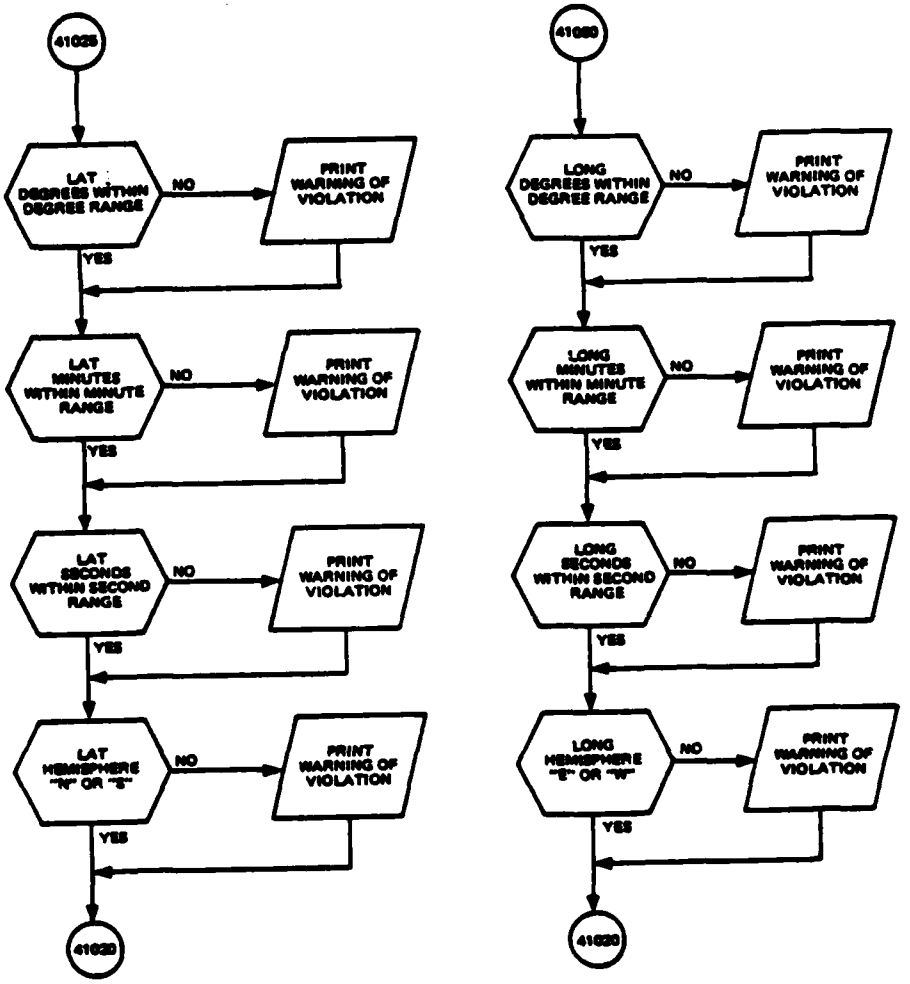


Figure 43.1. (Part 7 of 7)

THIS PAGE INTENTIONALLY LEFT BLANK

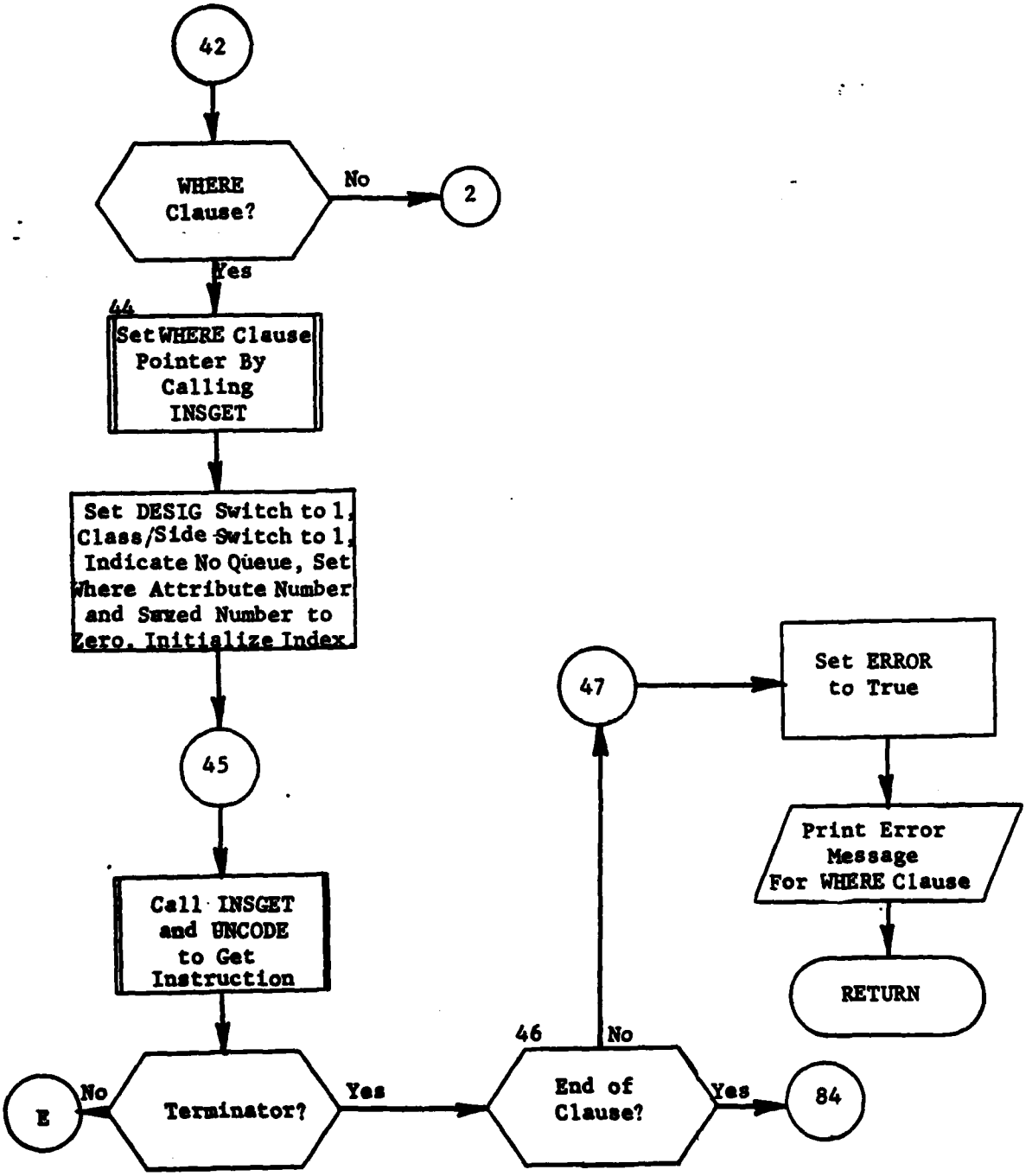


Figure 44. Subroutine CHANGE: Step Two (Part 1 of 9)

Step Three

The two attributes lists (ATNUMB and WHATNB) are now combined. Also it is determined if both clauses contained a queued set of attributes (see figure 45).

Step Four

The list of attributes (ATNUMB) is now used in an attempt to build a list of record types. The ATRIB chain is used to find the attributes in the list and it is determined for each attribute whether it is a single, multiple or control. For a single attribute, the record type it is on is added to the record type list (RTLIST). For multiple attributes (those in the SETTING clause) a list (MLTLST) is kept separately of the record types which contain them. For a control attribute the controlled record is added to the record type list (RTLIST). If the attribute appeared in the SETTING clause, the record type is also added to a special list (CTLLST). A separate list (CRECNM) is also made of the record types whose attributes appear in the SETTING clause as these are the types which will be changed (see figure 46).

Step Five

For each record type in the list (CTLLST) of controlled record types whose attribute was in the SETTING clause, PRIMHD is now called to determine their primary header. These headers are added to the list of record types (RTLIST) (see figure 47).

Step Six

Now the primary header is determined. If a value for CLASS was included, it is used to do this. If no such value was included the record type with the highest number is used. PRIMHD is called to determine the primary header. The primary chains down from the primary header are now checked against the multiple attribute record list. Any found are included in the record type list (RTLIST) (see figure 48).

Step Seven

First LINKUP is called to complete the record type list (RTLIST). SETSCH is now called to build the retrieval scheme. The retrieval scheme is used to determine the retrieval order of the record types which are to be changed and they are placed in the CHORD list in this order (see figure 49).

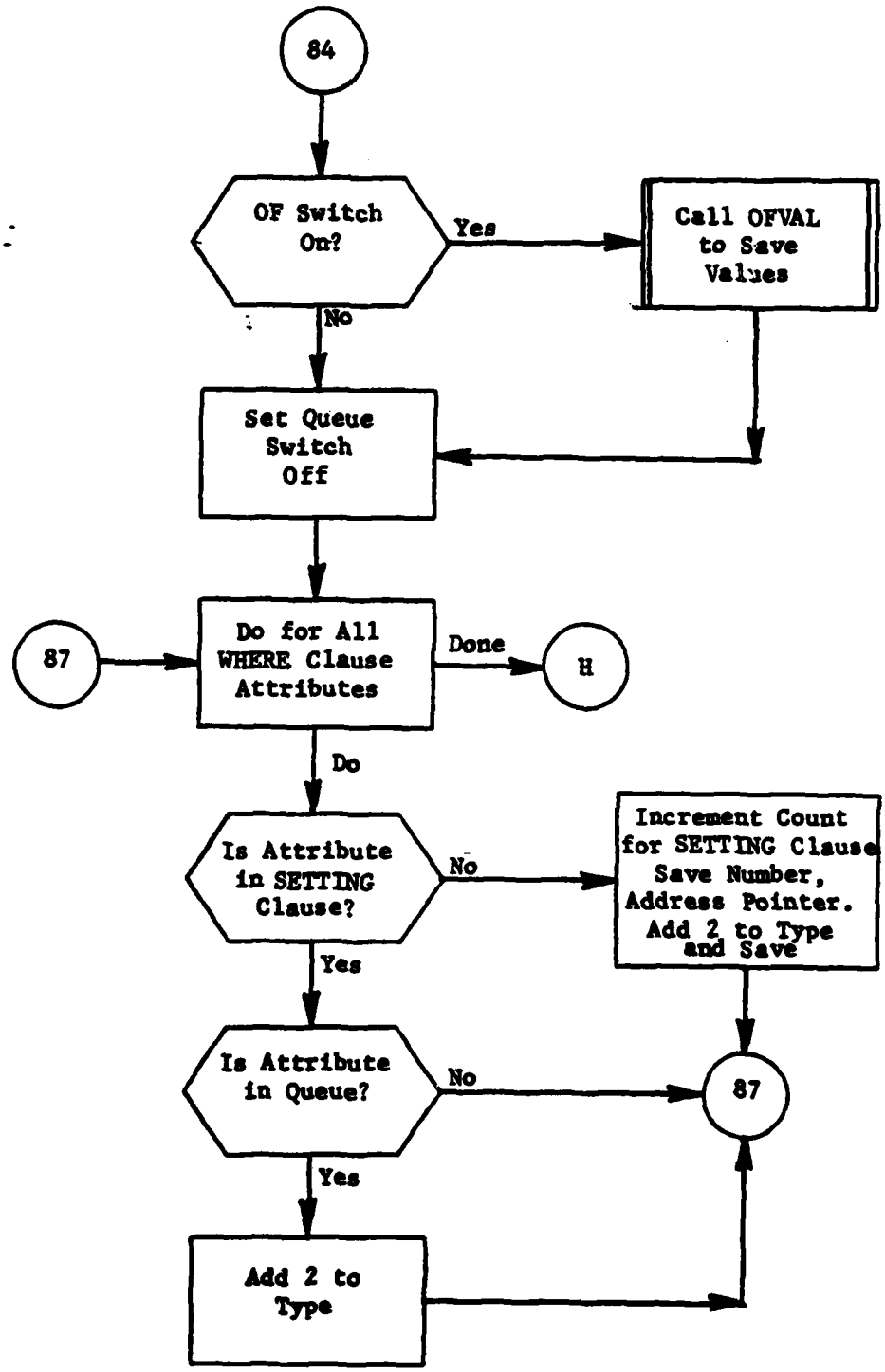


Figure 45. Subroutine CHANGE: Step Three (Part 1 of 2)

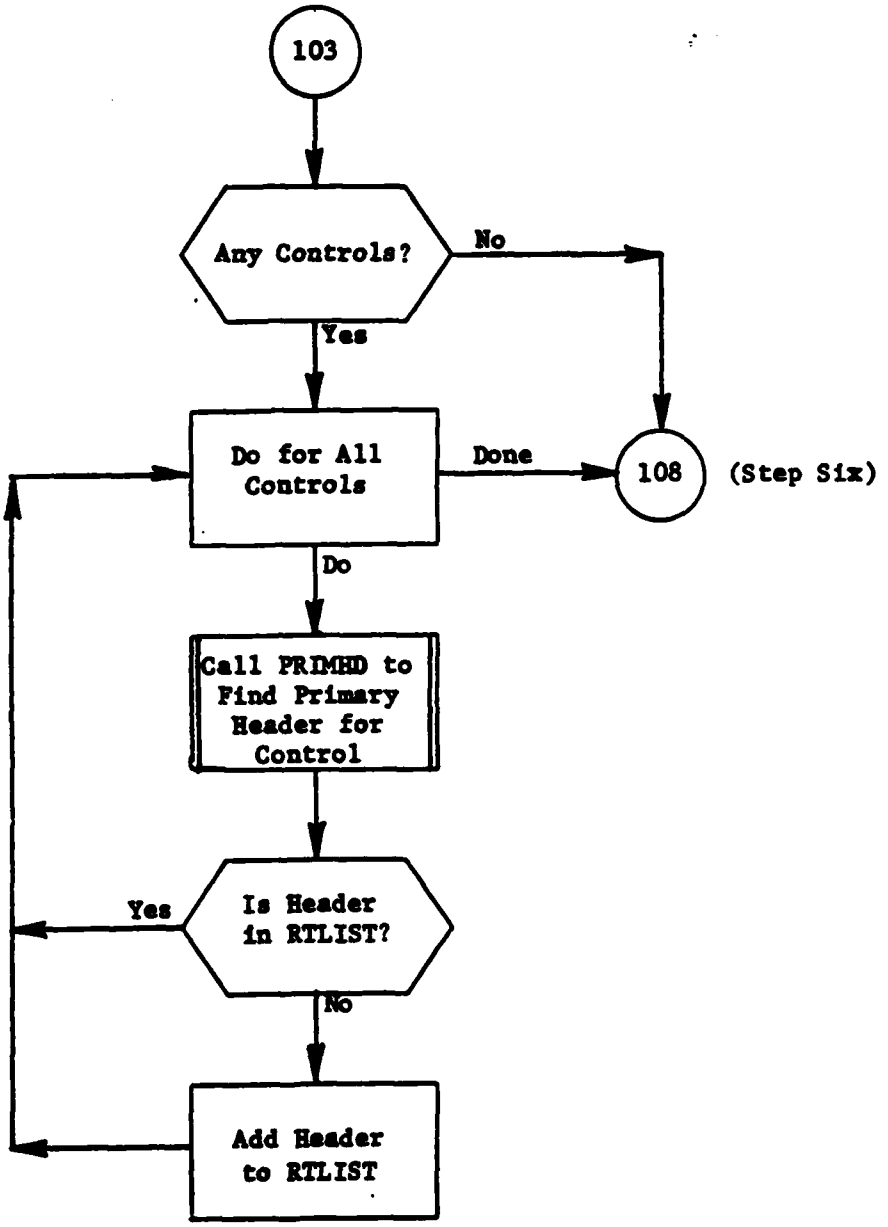


Figure 47. Subroutine CHANGE: Step Five

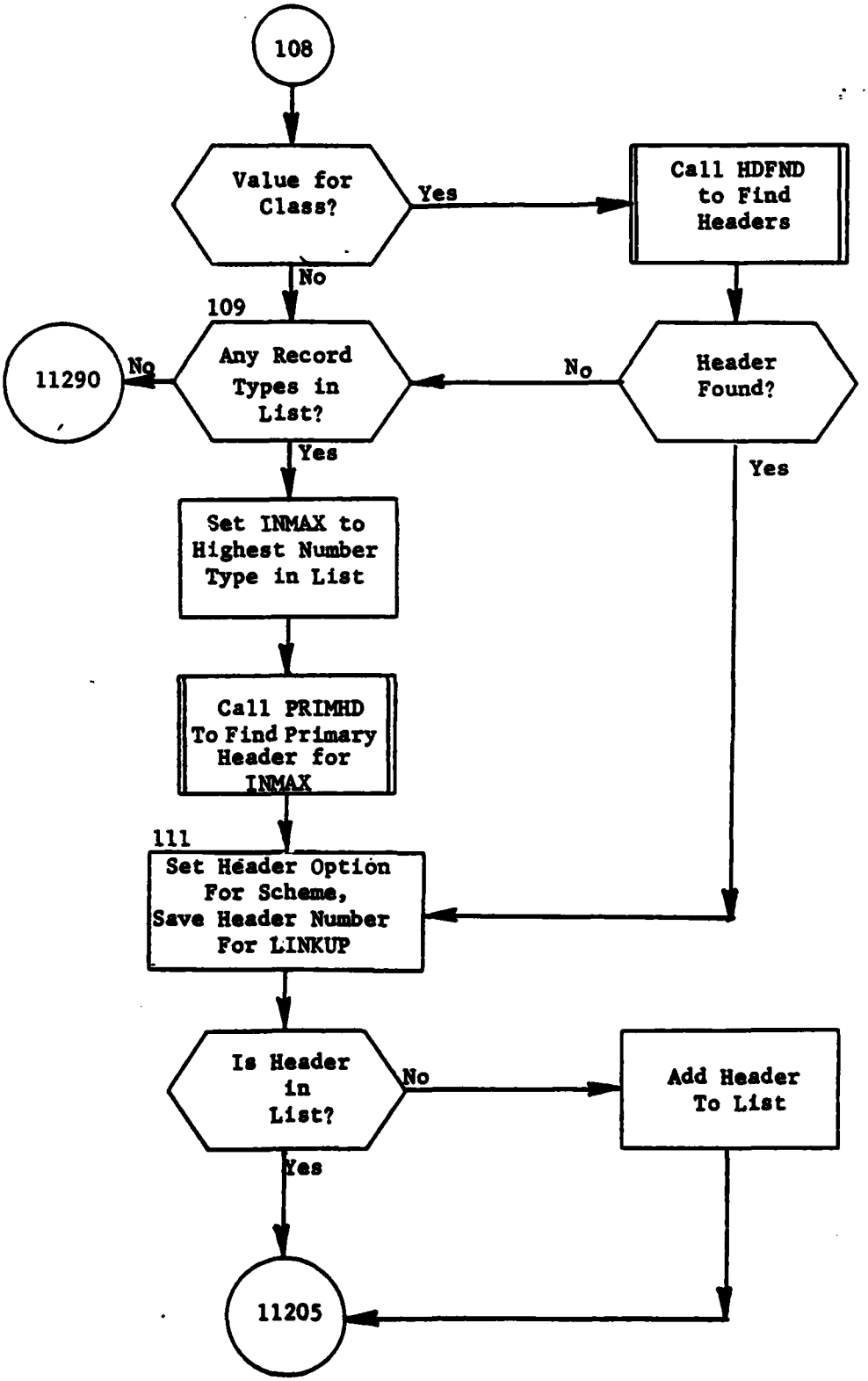


Figure 48. Subroutine CHANGE: Step Six (Part 1 of 4)

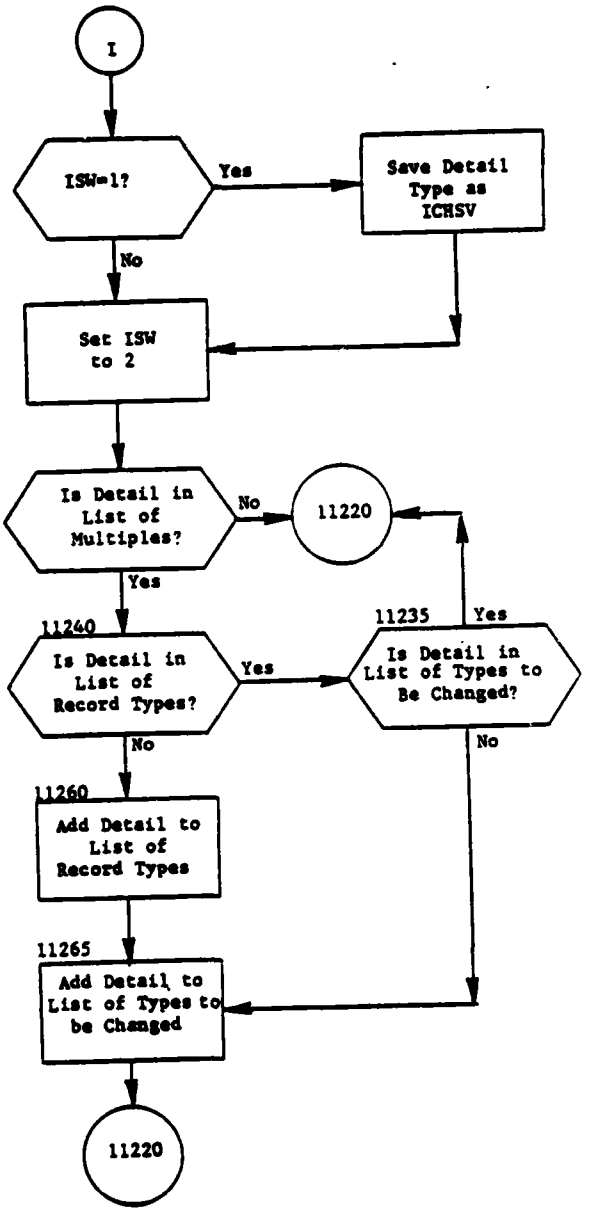


Figure 48. (Part 4 of 4)

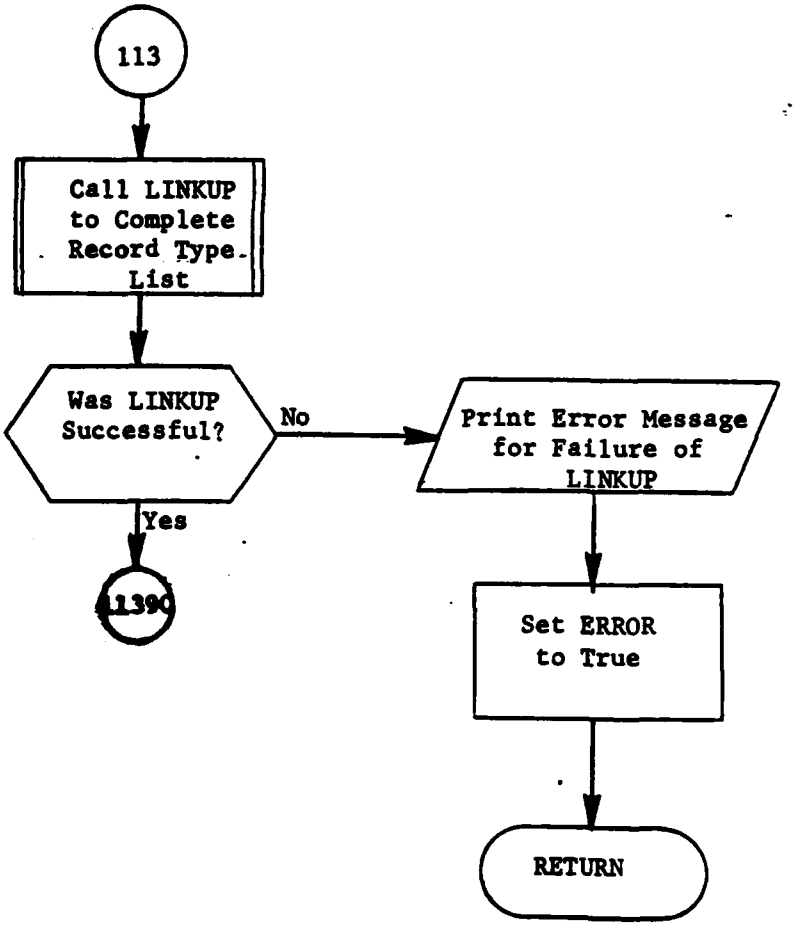


Figure 49. Subroutine CHANGE: Step Seven (Part 1 of 5)

Step Eight

First, if the WHERE clause contained only the DESIG attribute, DESSCH is called to build a DESIG driven retrieval scheme (see section 4.8.1). Then the queue of DESIG values is processed one at a time. For each value NXTDES is called to retrieve the appropriate record. Then all SETTING clause attributes are set to their input values. The queued attributes are set to the values whose position in the queue correspond to the position of the DESIG value in its queue. Then the record types in the CHORD array have MODFY called for them in the order they appear.

If the retrieval is not DESIG driven the process is similar. The GETNXT routine is called until it passes back the indication that the retrieval process is over (ISW=2). For each record retrieved, XWHERE is called and if the record is one of those desired, the attribute values are set. If there is a queue, the WHERE is searched for the appropriate match and the corresponding SETTING queue values are used. Then MODFY is called for the elements of the CHORD array (see figure 50).

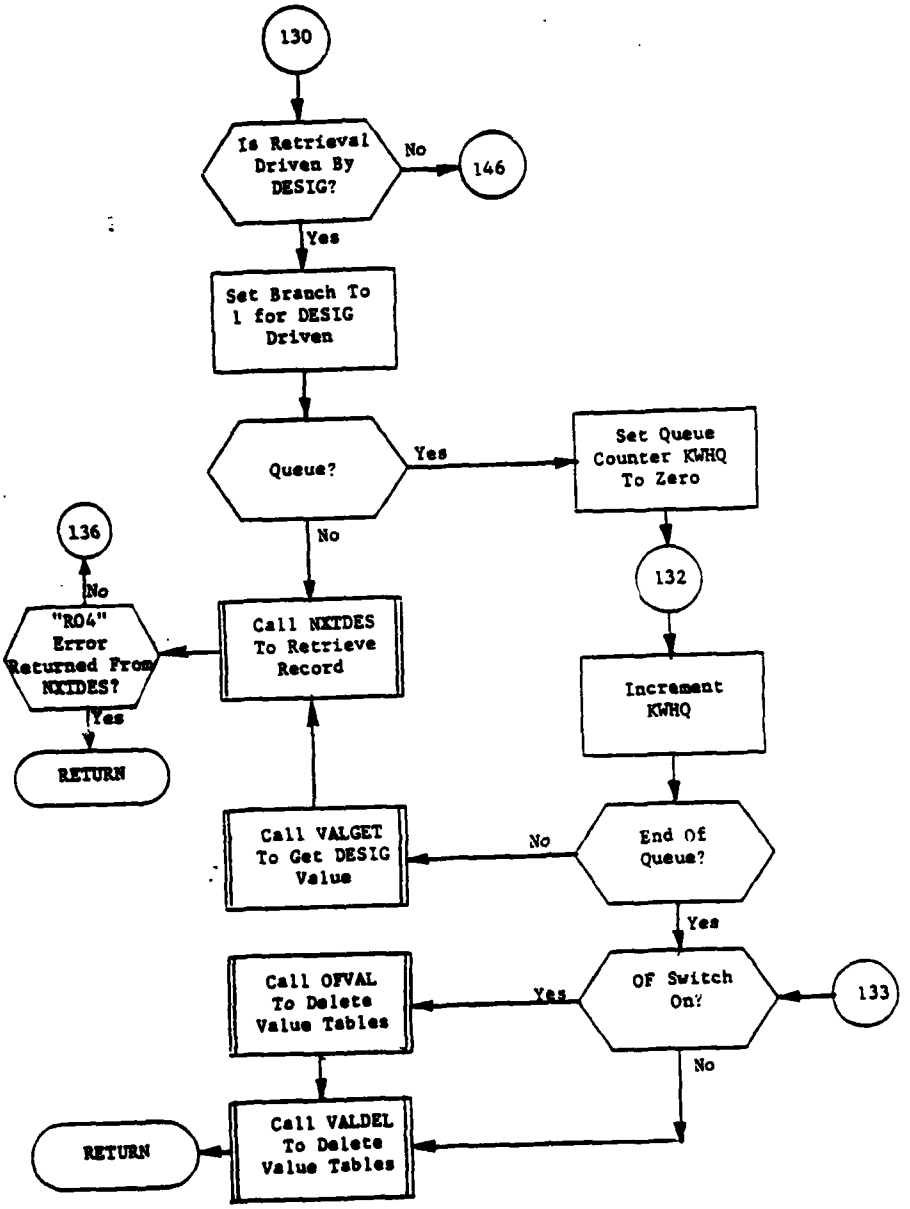


Figure 50. Subroutine CHANGE: Step Eight (Part 1 of 7)

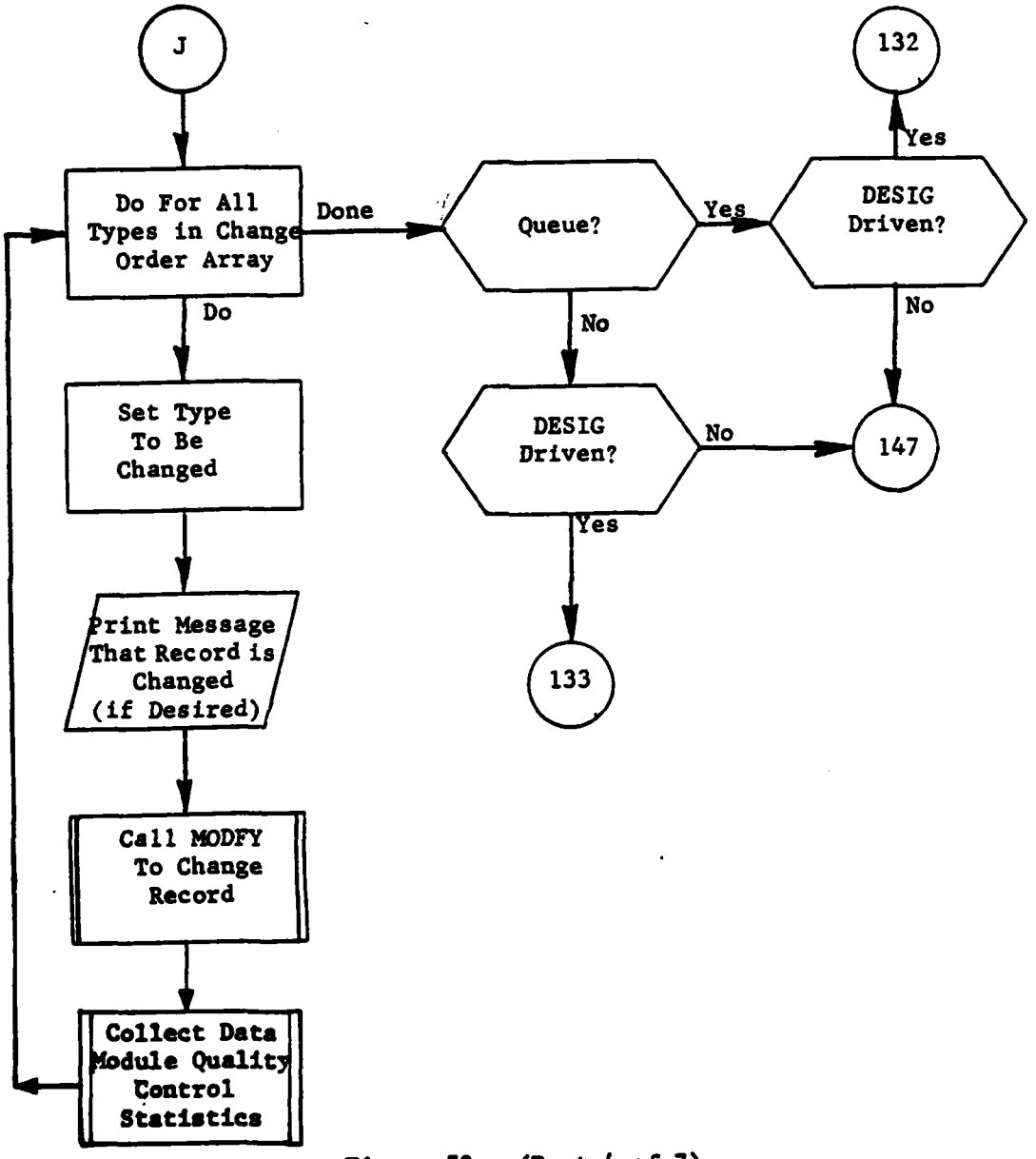


Figure 50. (Part 4 of 7)

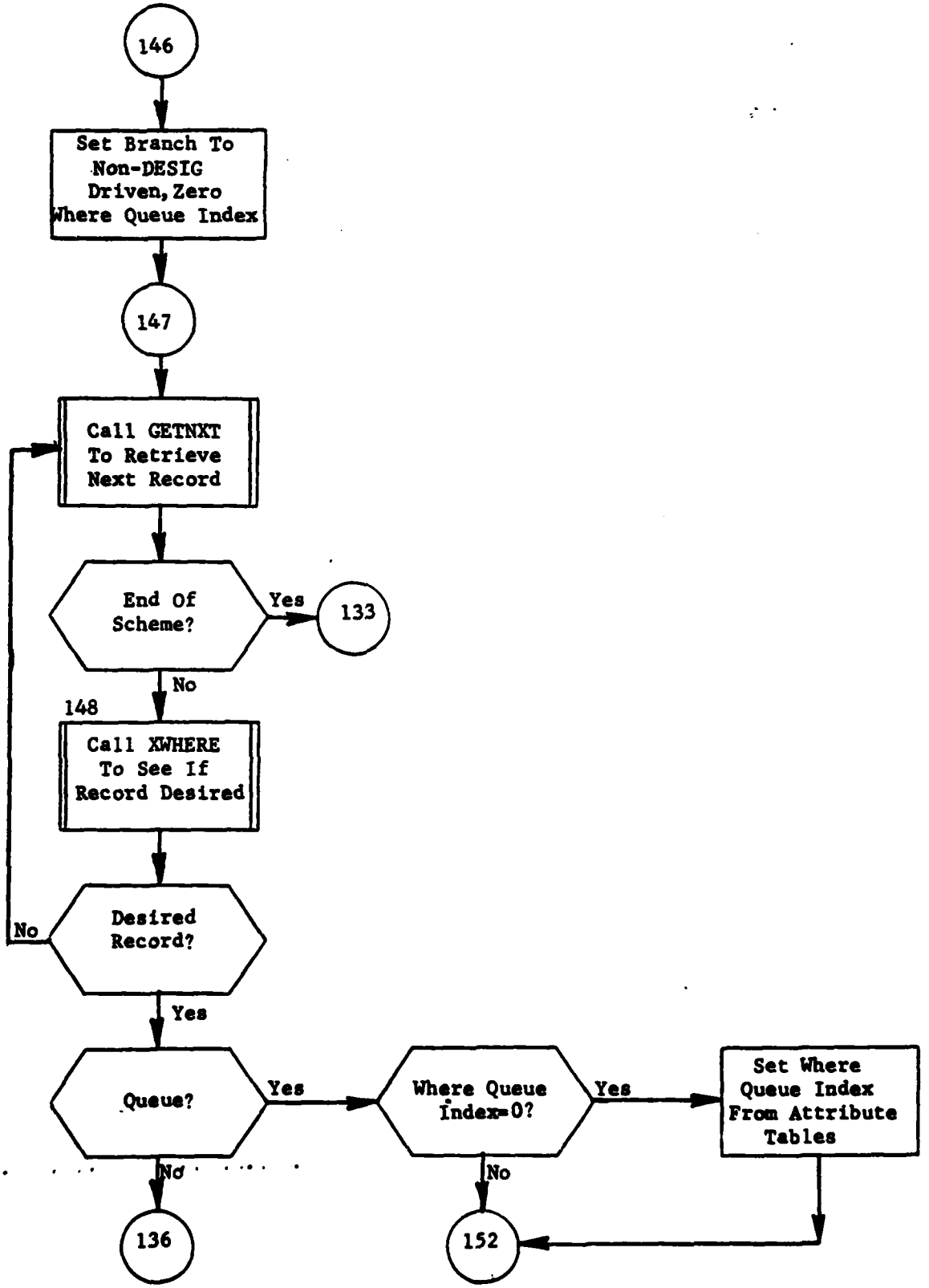


Figure 50. (Part 5 of 7)

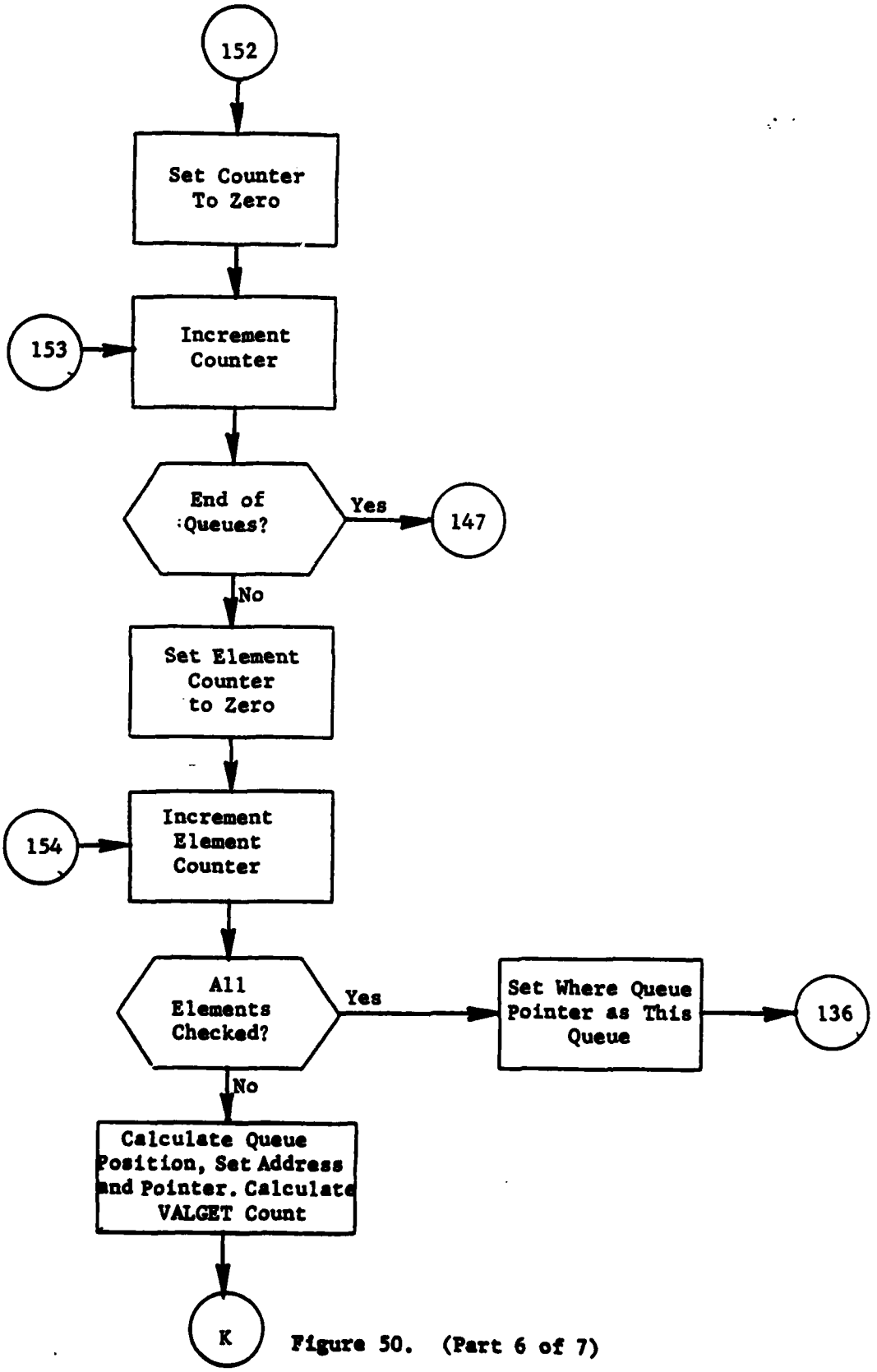


Figure 50. (Part 6 of 7)

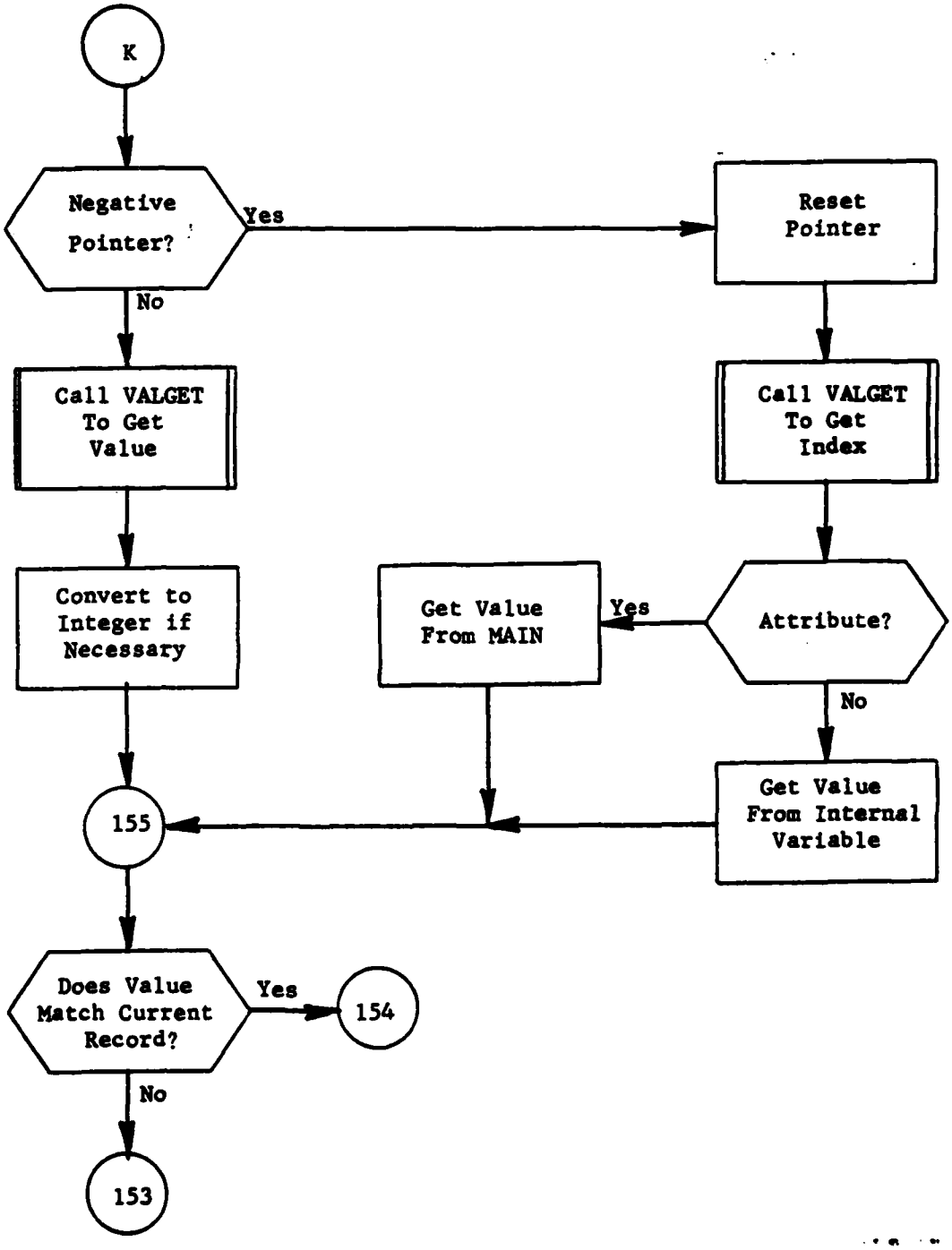


Figure 50. (Part 7 of 7)

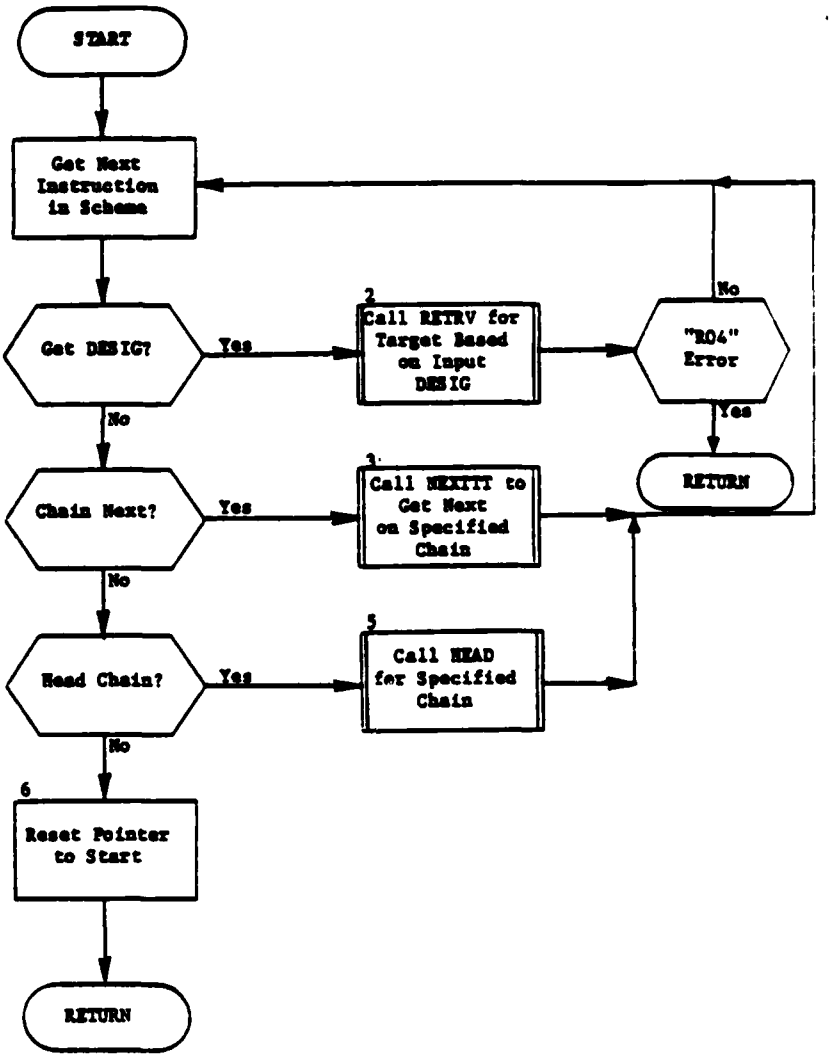


Figure 52. Subroutine NXTDES

4.9 Subroutine CREAAT

PURPOSE: To create new data records

ENTRY POINTS: CREAAT

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C15, C20, C30, ERRCOM, OOPS, ORDER, PRINSP, SCHEME, QC

SUBROUTINES CALLED: DIRECT, FNDTAR, GETNXT, GETTAR, HDFND, HEAD, INSGET, LINKUP, NEXTTT, OFVAL, PRIMHD, RETRV, SETSCH, STORE, UNCODE, VALDEL, VALFND, VALGET, VALPUT, XLL, XMATH

CALLED BY: ENTMOD (DATA)

Method:

The CREAAT verb process may be broken into 15 steps. Of these, step 1 is executed but once and controls as many executions of steps 2 through 14 as there are SETTING clauses. Furthermore, step 13 is executed after the first execution of step 14 and 15 for a SETTING clause. Thereafter steps 13, 14, and 15 are executed for every unique combination of any data queues (see figure 39)

Step One

First the error routine ERPROC is set to accept duplicate records. Then the input is scanned for the presence of SUPRESSING and SAME adverbs. If a SUPRESSING adverb is found, the DRCTSW switch which is originally set to true is set to false to indicate that input values should not be edited. If a SAME adverb is found the SAMESW switch is set to true and the identifying attribute's number, address and value are saved in ISMIDN, ISMIDA, and SMVAL, respectively. Now each SETTING adverb is processed in the order input with steps 2 through 14 being executed for each. When all SETTING adverbs have been processed the subroutine exits (see figure 53).

Step Two

The current SETTING clause is scanned. The limits of any mathematical calculations are found for XMATH and OF phrases which are involved in those calculations are resolved using VALFND and their values saved with OFVAL. Attributes which are encountered are placed in a list (ATNUMB) and counts are made of those which appear more than once or in collections (see figure 54).

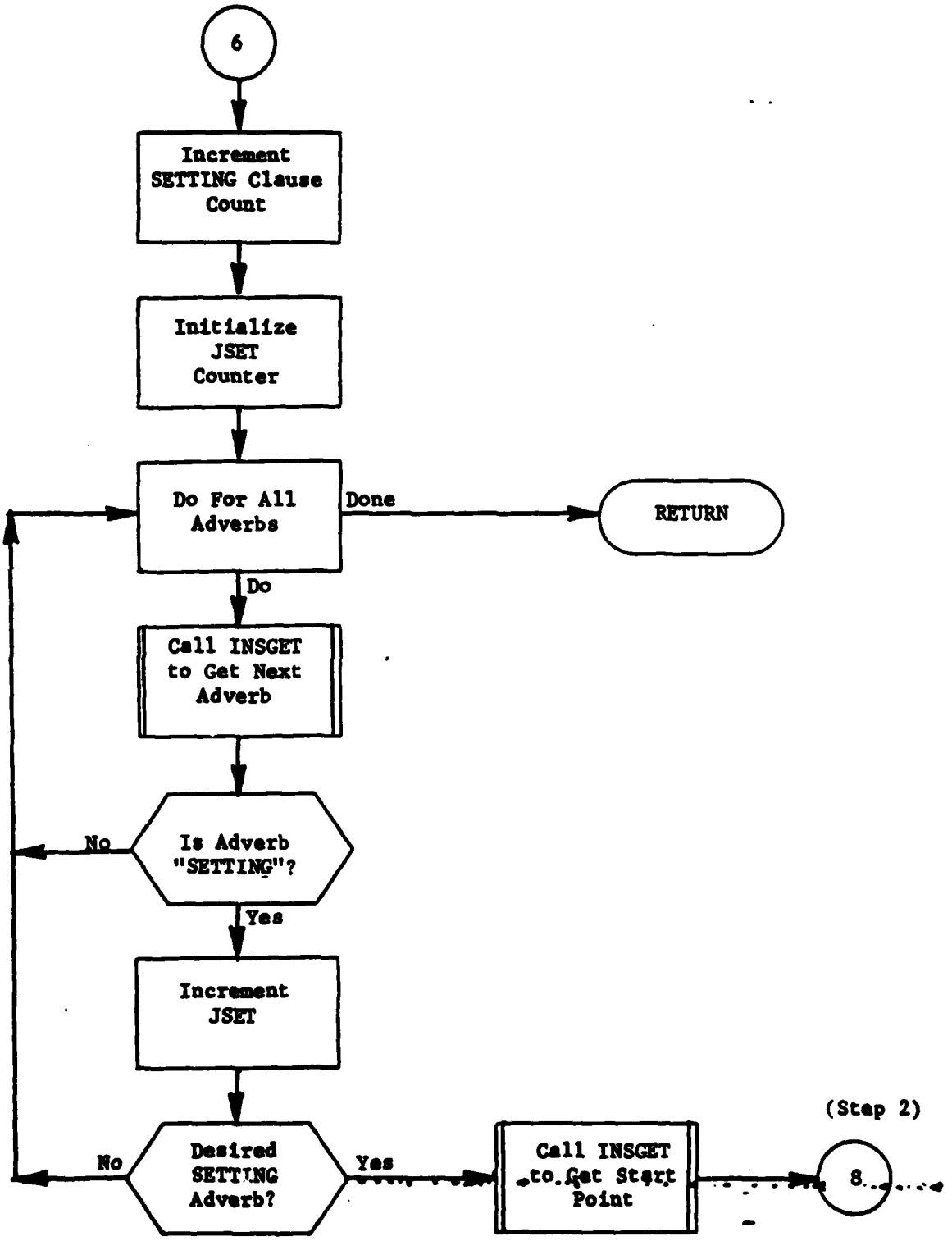


Figure 53. (Part 3 of 3)

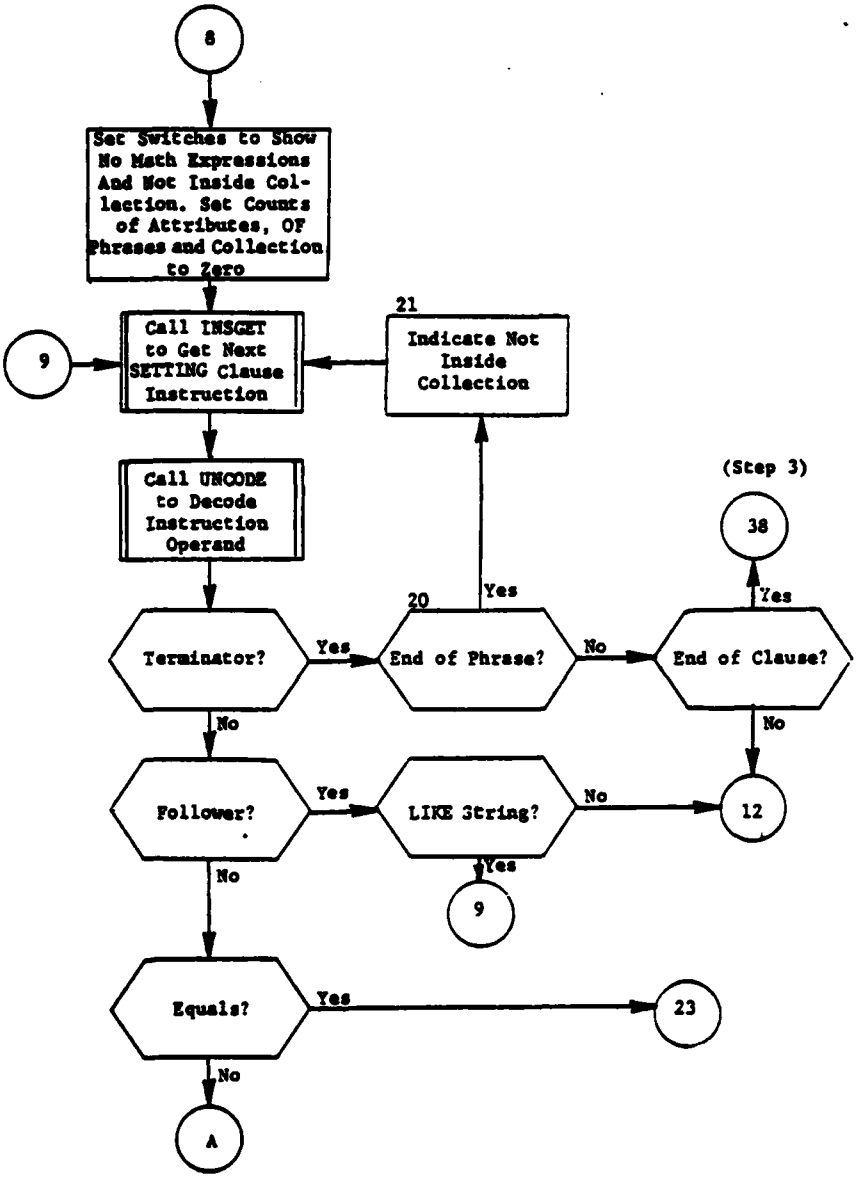


Figure 54. Subroutine CREAAT: Step 2 (Part 1 of 6)

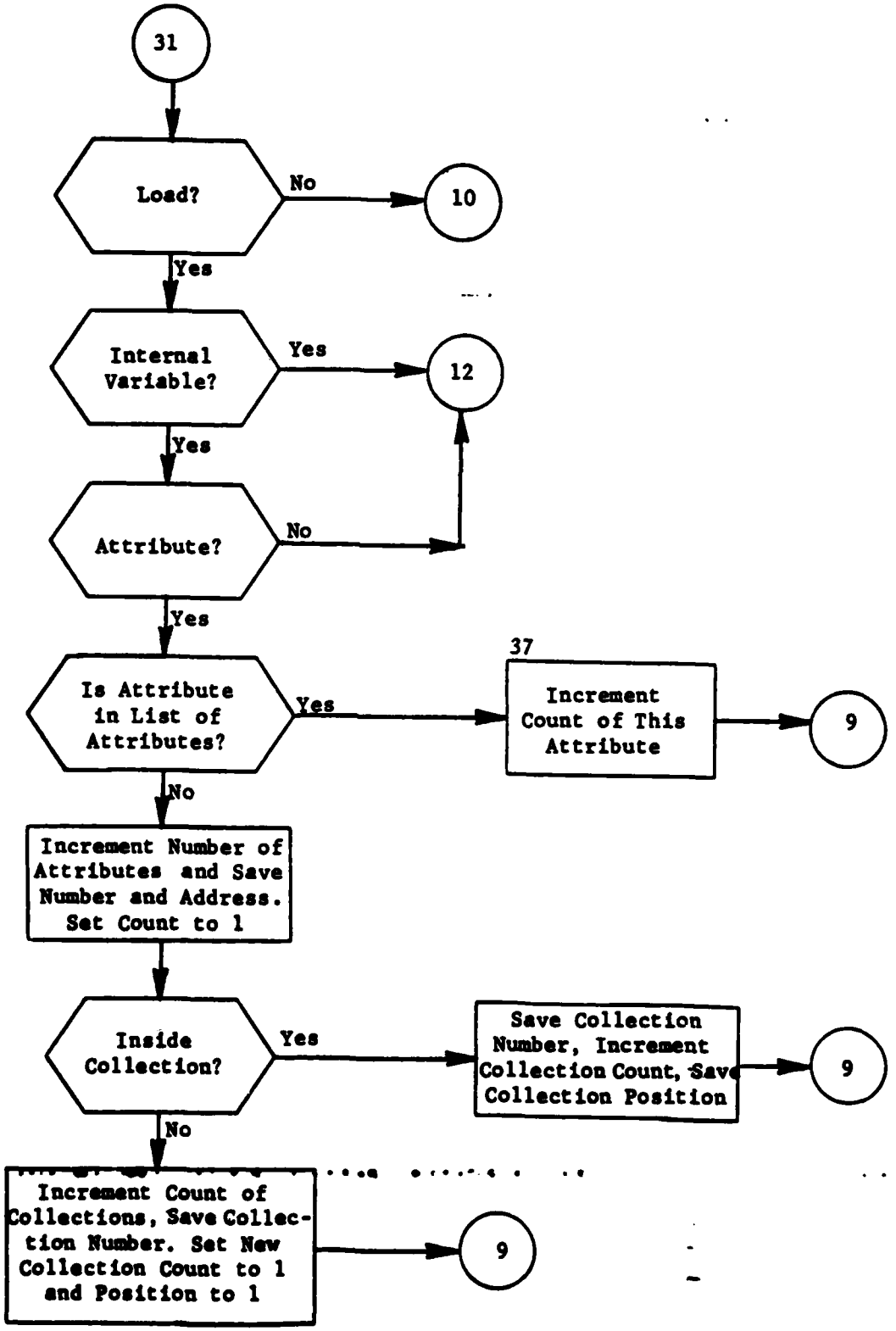


Figure 54. (Part 6 of 6)

Step Three

The list of attributes (ATNUMB) is processed to set up value storage. Those which occurred only once are given a pointer to the VALBUF array. Those which occurred a number of times or in collections are given a position in a queue maintained through VALPUT (see figure 55).

Step Four

The SETTING clause is scanned again. This time the values encountered are stored according to conditions set in step three. If an attribute is set via a LIKE phrase or equal to an attribute with an OF phrase, VALFND is called to obtain the desired value. If an attribute is set equal to a calculation, XMATH is called for the records obtained when VALFND is used with the SAME clause values as identifiers. If an attribute is set equal to another attribute which does not have an OF phrase, the SAME clause is also used as the identifier. In either case where the SAME clause values are used an error condition results if no SAME clause was provided (see figure 56).

Step Five

The ATRIB chain is now used with two objectives. First as each attribute in the list of attributes (ATNUMB) is found on the ATRIB chain, a list (RTLIST) is made of the record types which contain them. Single defined attributes record types are added to the list immediately. Multiple and control attributes are kept in separate lists and resolved later. Furthermore, unless DRCTSW is false, the value or values assigned to each attribute are checked against the directory, except LAT and LONG which are compared against hard-coded values. Any values which violate the directory are noted (see figure 57).

Step Six

Control attributes are now checked to see if they already have their record types in the main list (RTLIST) (which up to now contains only record types from single attributes). If the uncontrolled record type (CNTB) is not in the list the controlled record type (CNTA) is added to the list. Also, any multiply defined attribute has its list (MLATPT) of record types compared to the main list (RTLIST). If any match is found, the attribute is considered resolved (see figure 58).

Step Seven

The primary header is now found in one of two ways. If a value was included for the CLASS attribute this value is used in a call to HDFND. If not, the record type which has had the most attributes set is used in a call to PRIMHD. An attempt is now made to resolve any remaining multiple attributes by searching on chains down from the primary header. Any record types on these chains which appear in the multiple attribute lists are added to the main list (see figure 59).

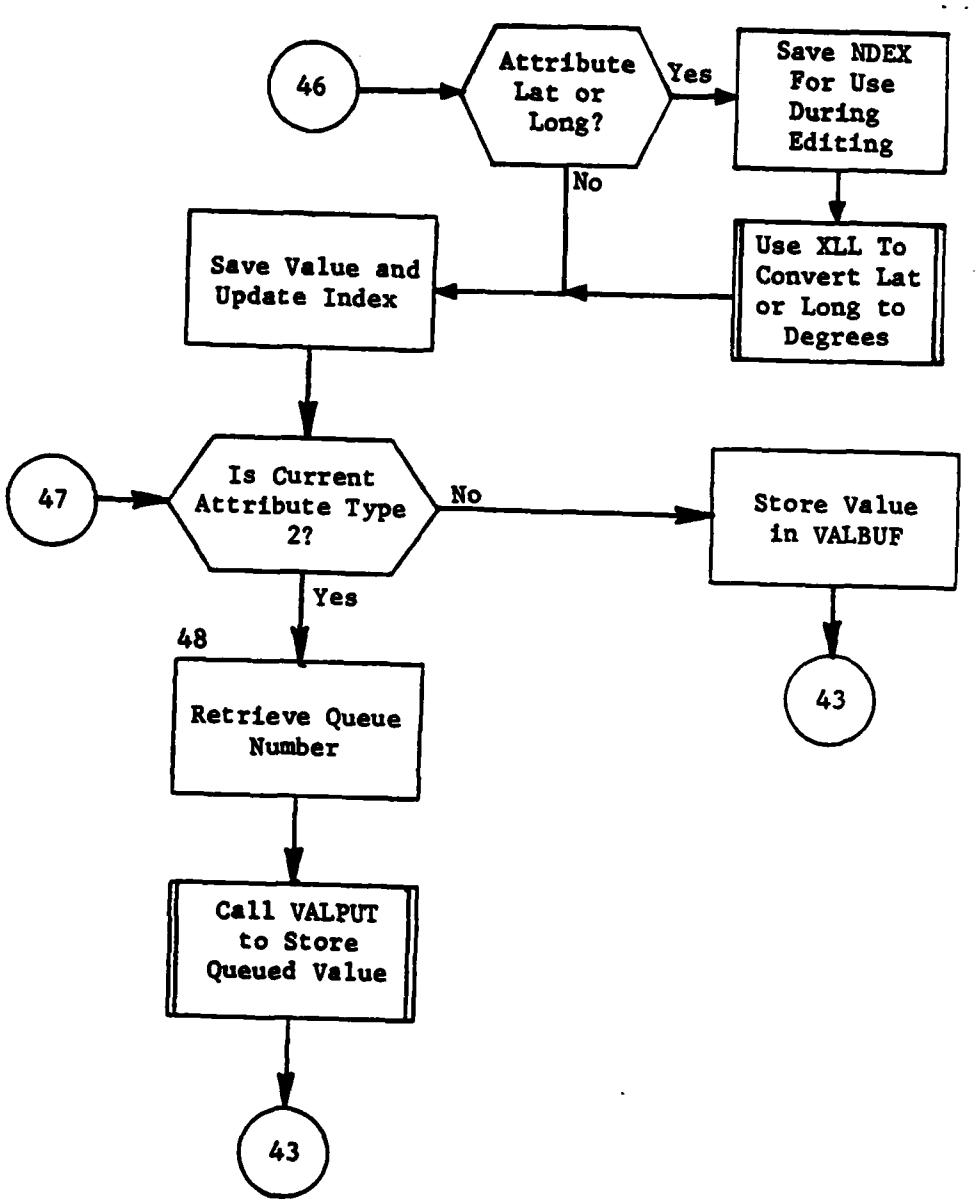


Figure 56. (Part 3 of 5)

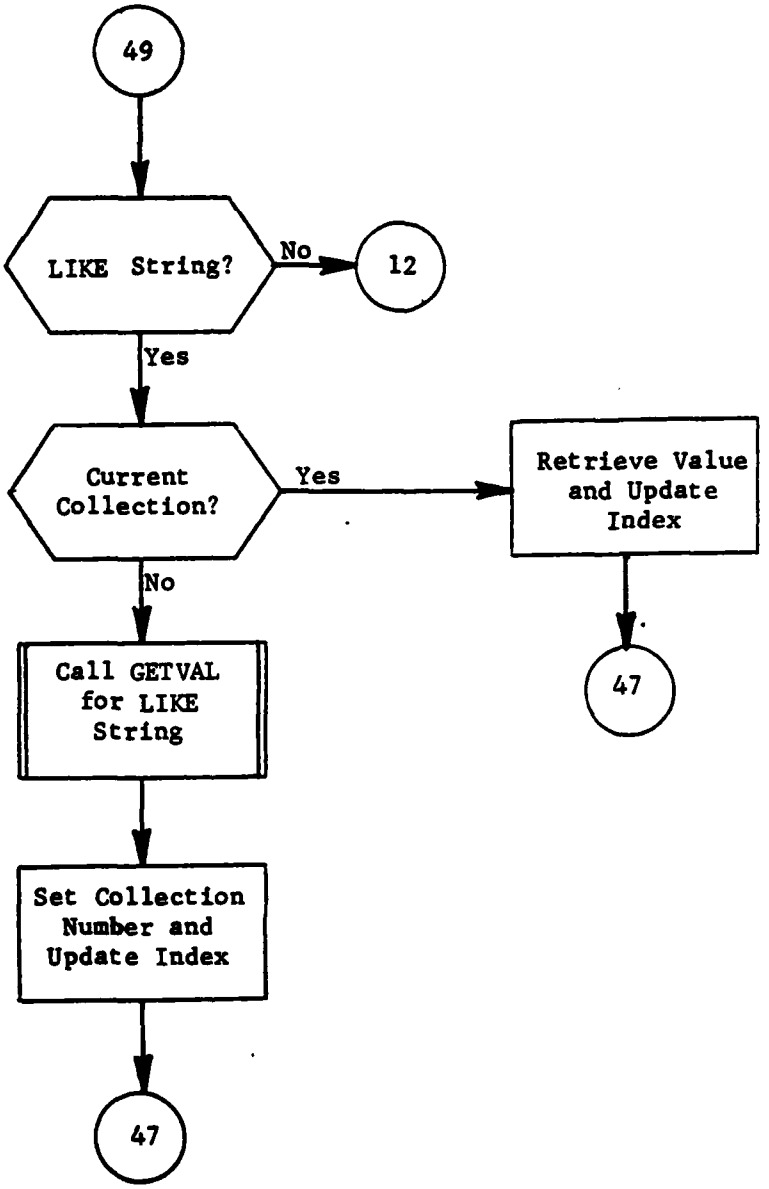


Figure 56. (Part 4 of 5)

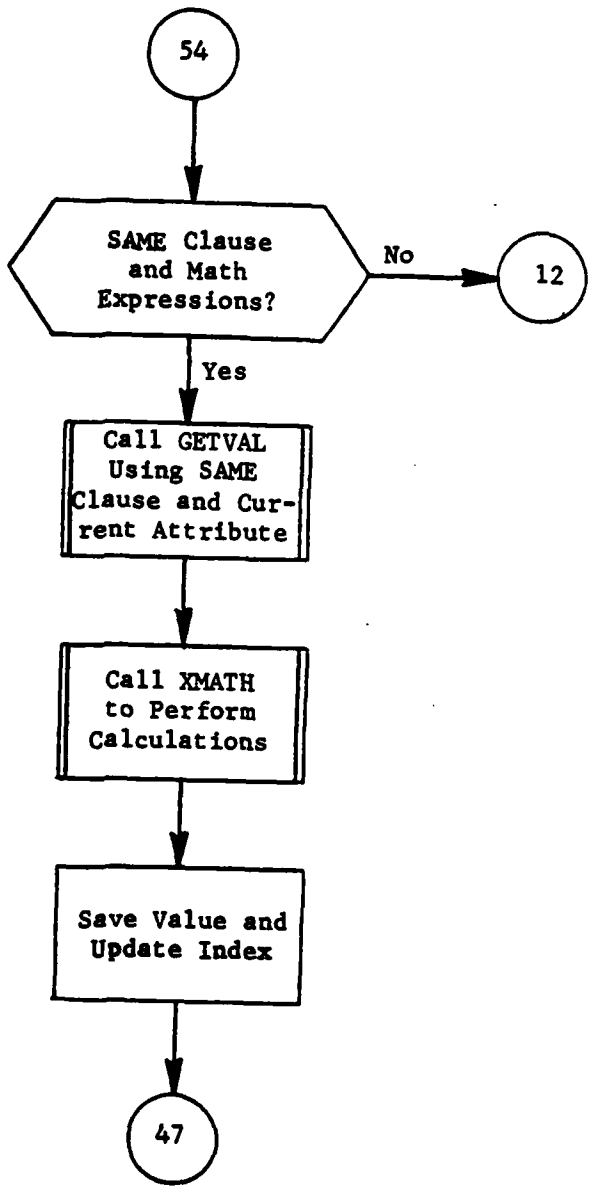


Figure 56. (Part 5 of 5)

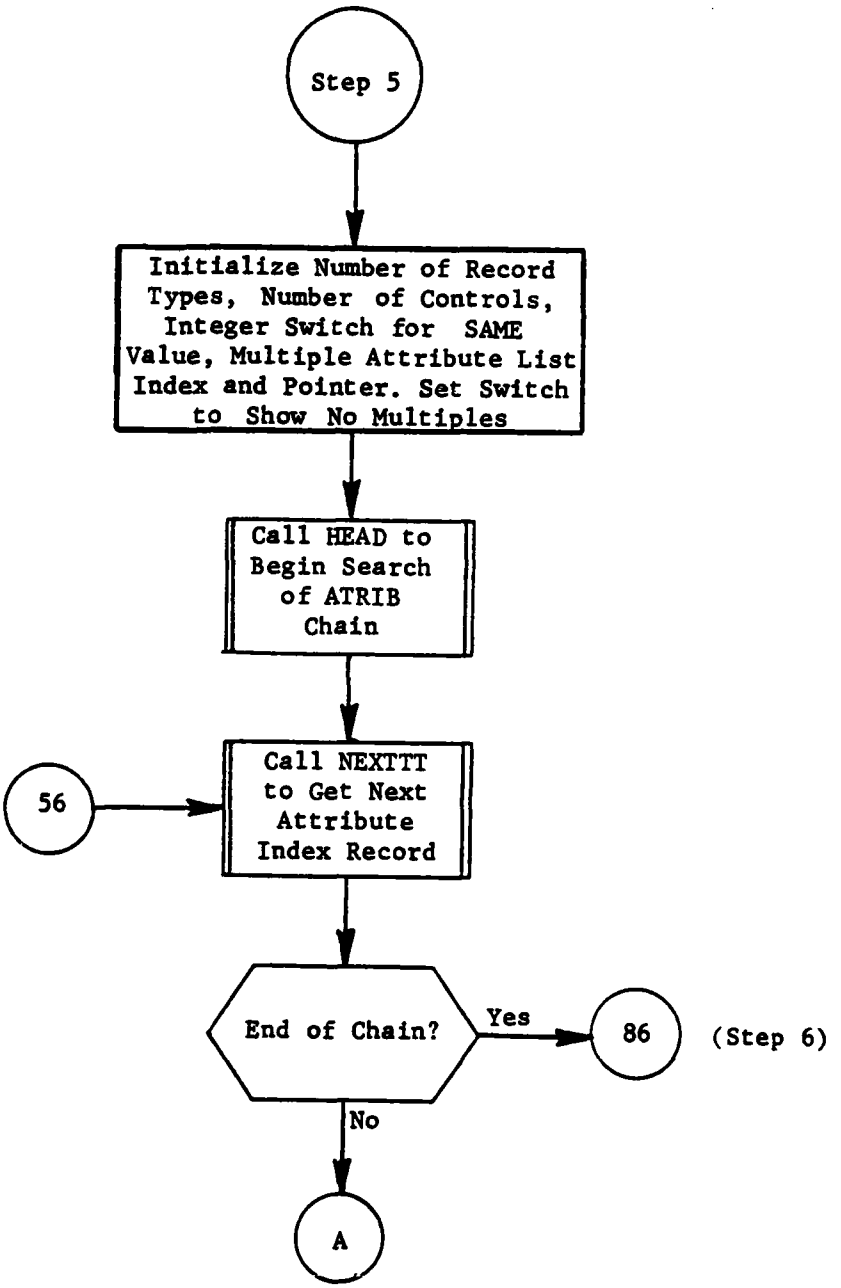


Figure 57. Subroutine CREAAT: Step 5 (Part 1 of 8)

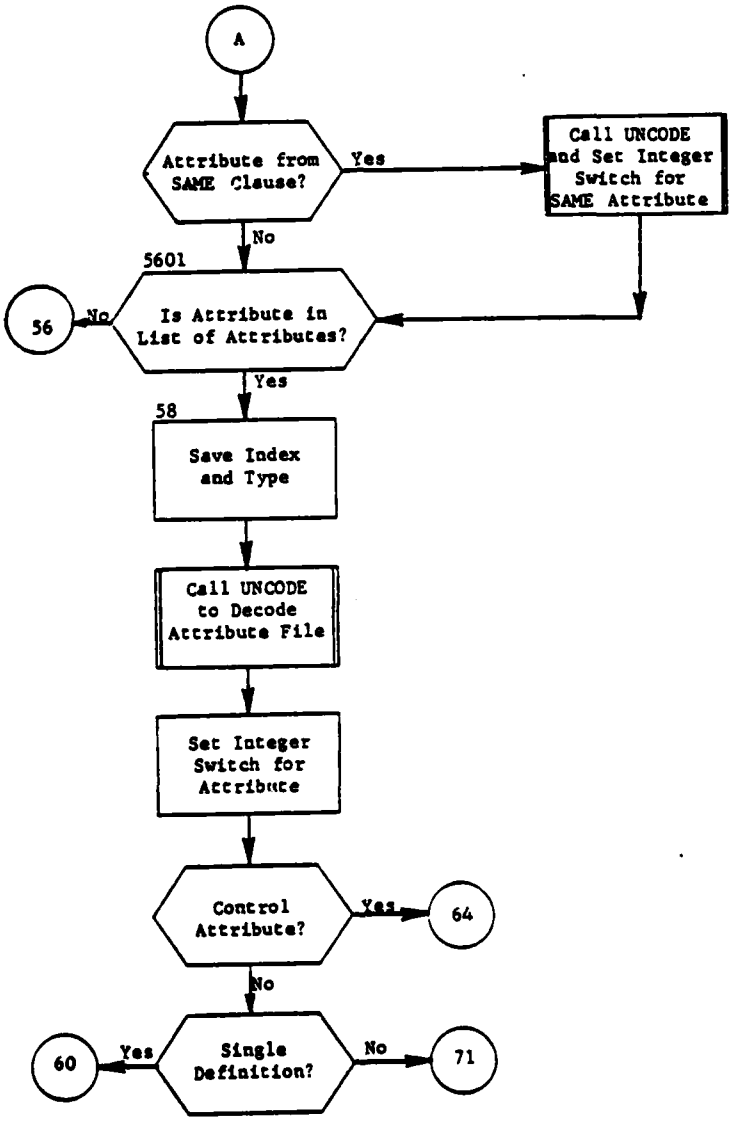


Figure 57. (Part 2 of 8)

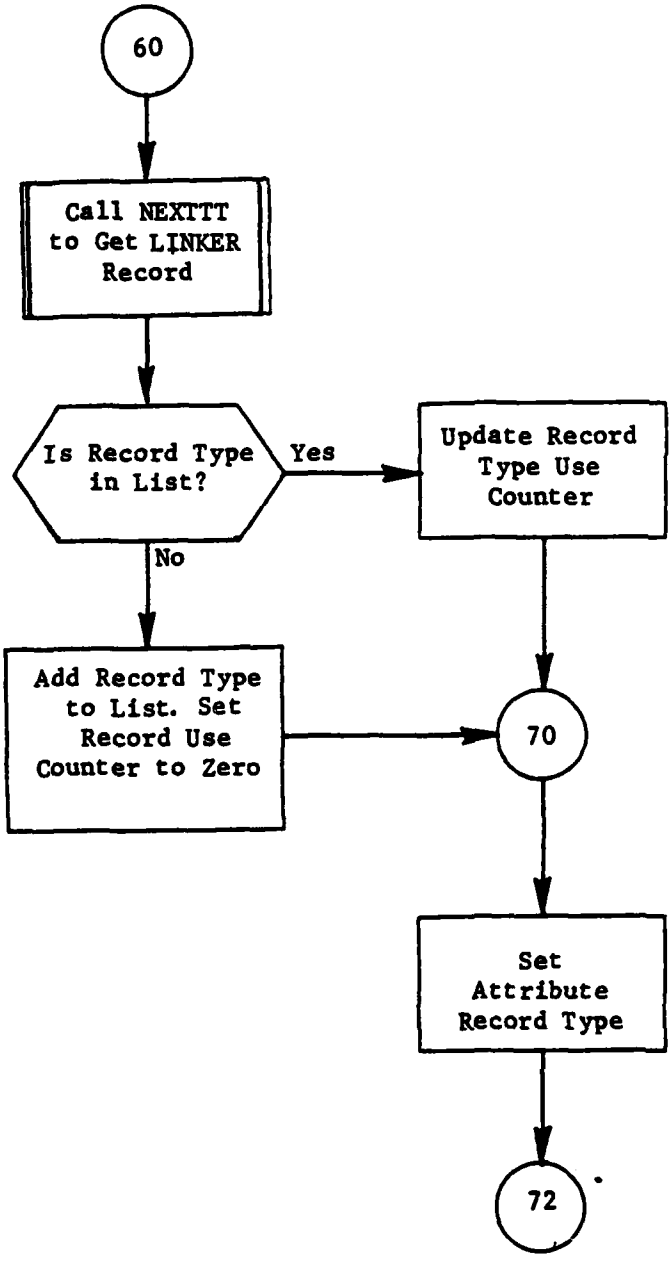


Figure 57. (Part 3 of 8)

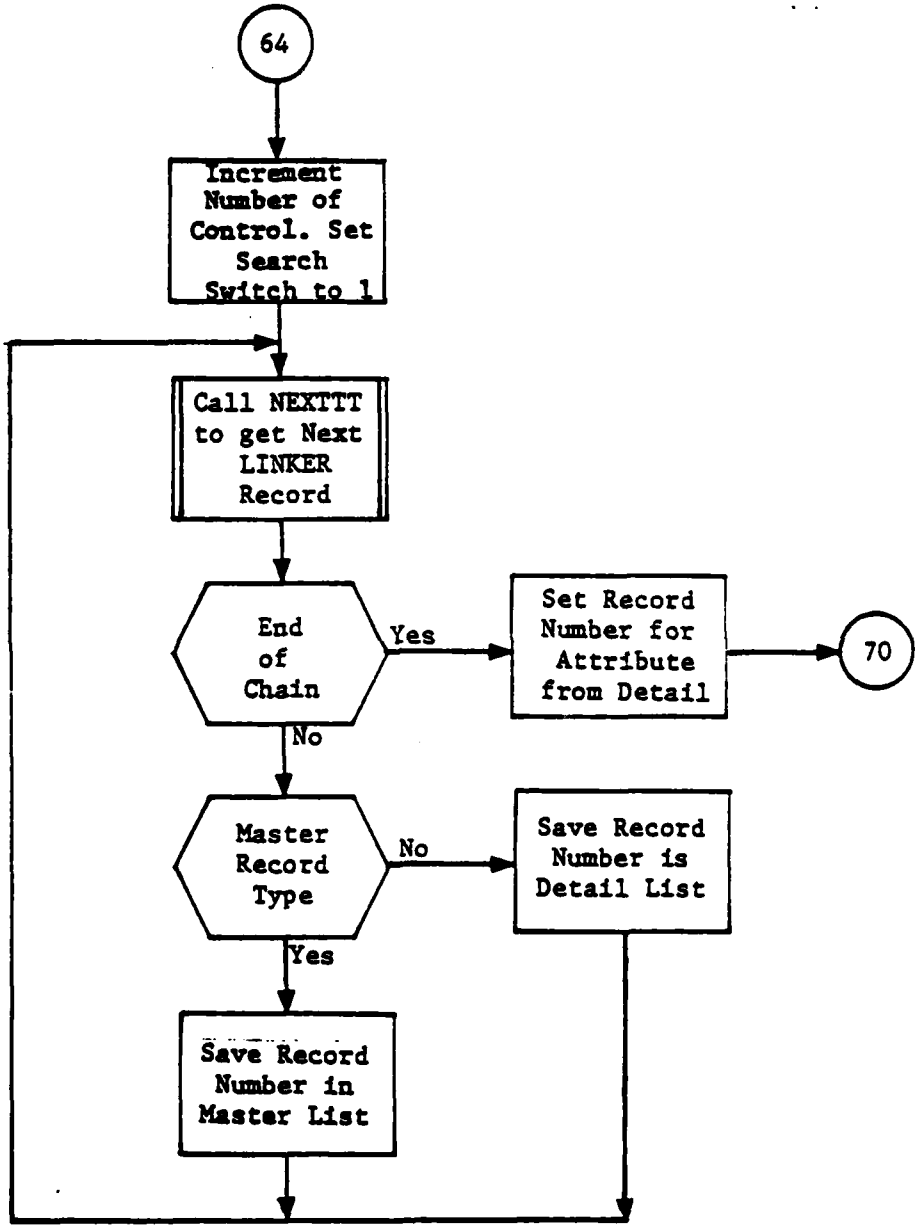


Figure 57. (Part 4 of 8)

THIS PAGE INTENTIONALLY LEFT BLANK

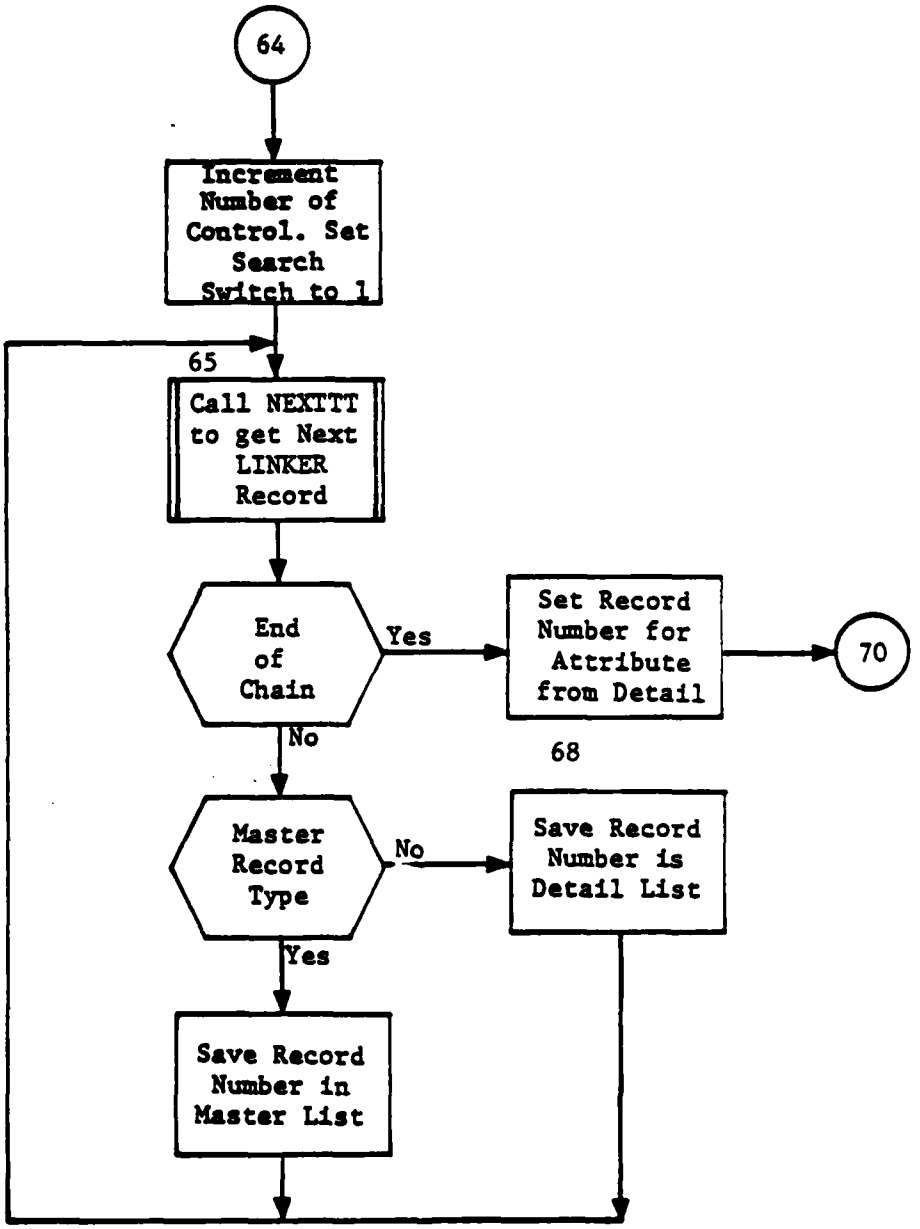


Figure 57. (Part 4 of 8)

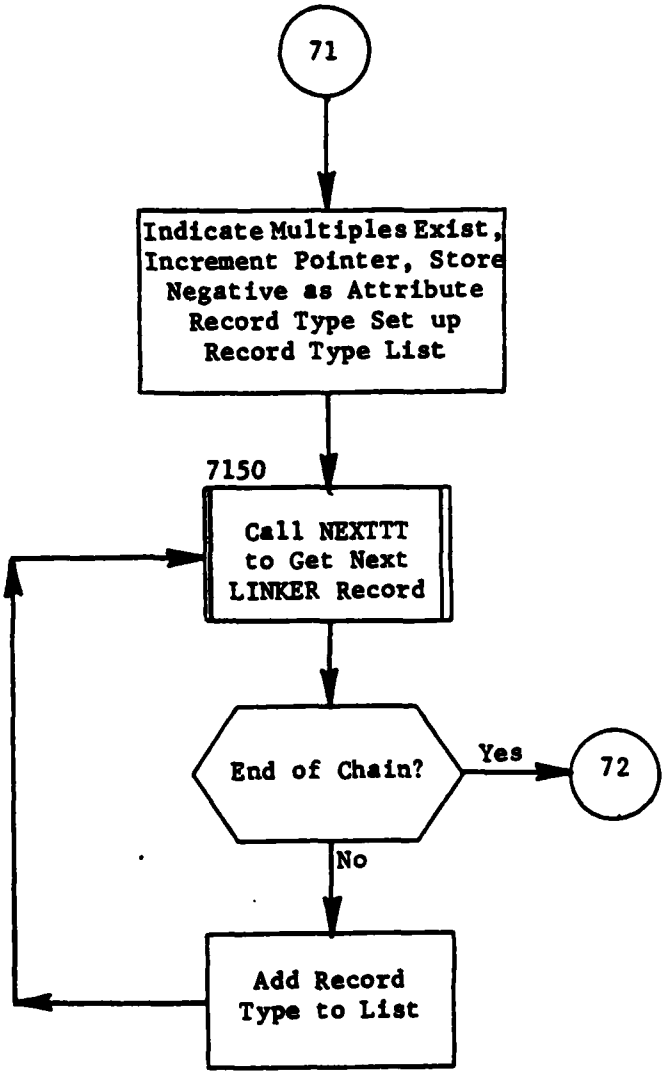


Figure 57. (Part 5 of 8)

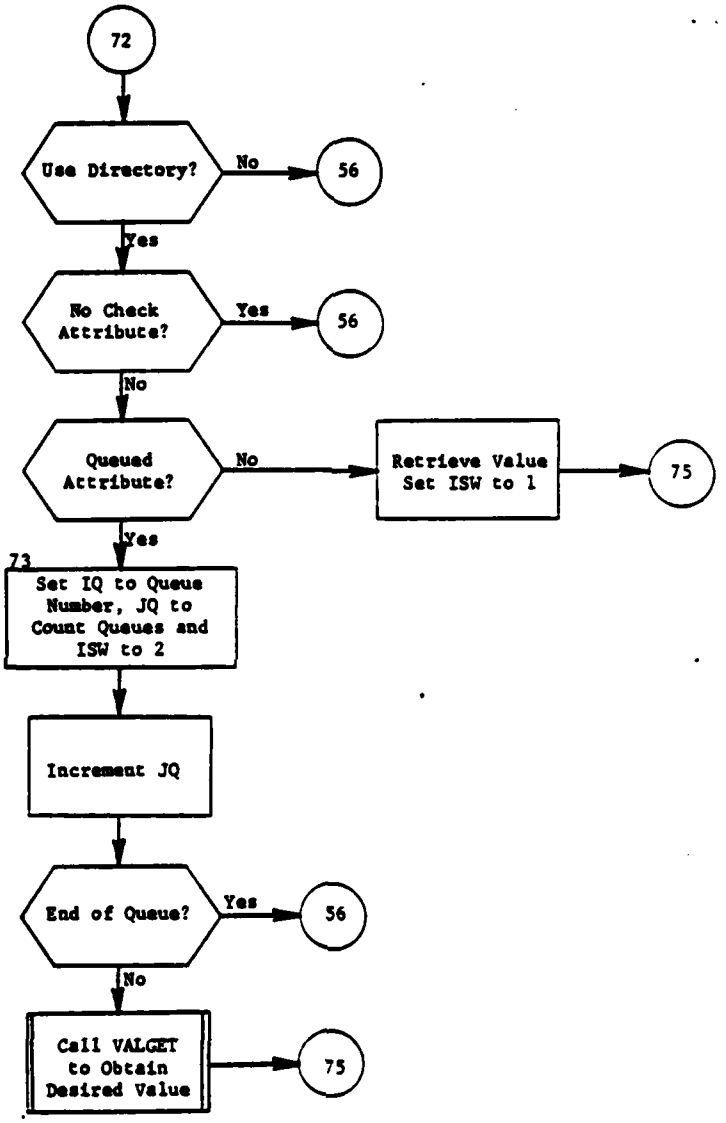


Figure 57. (Part 6 of 8)

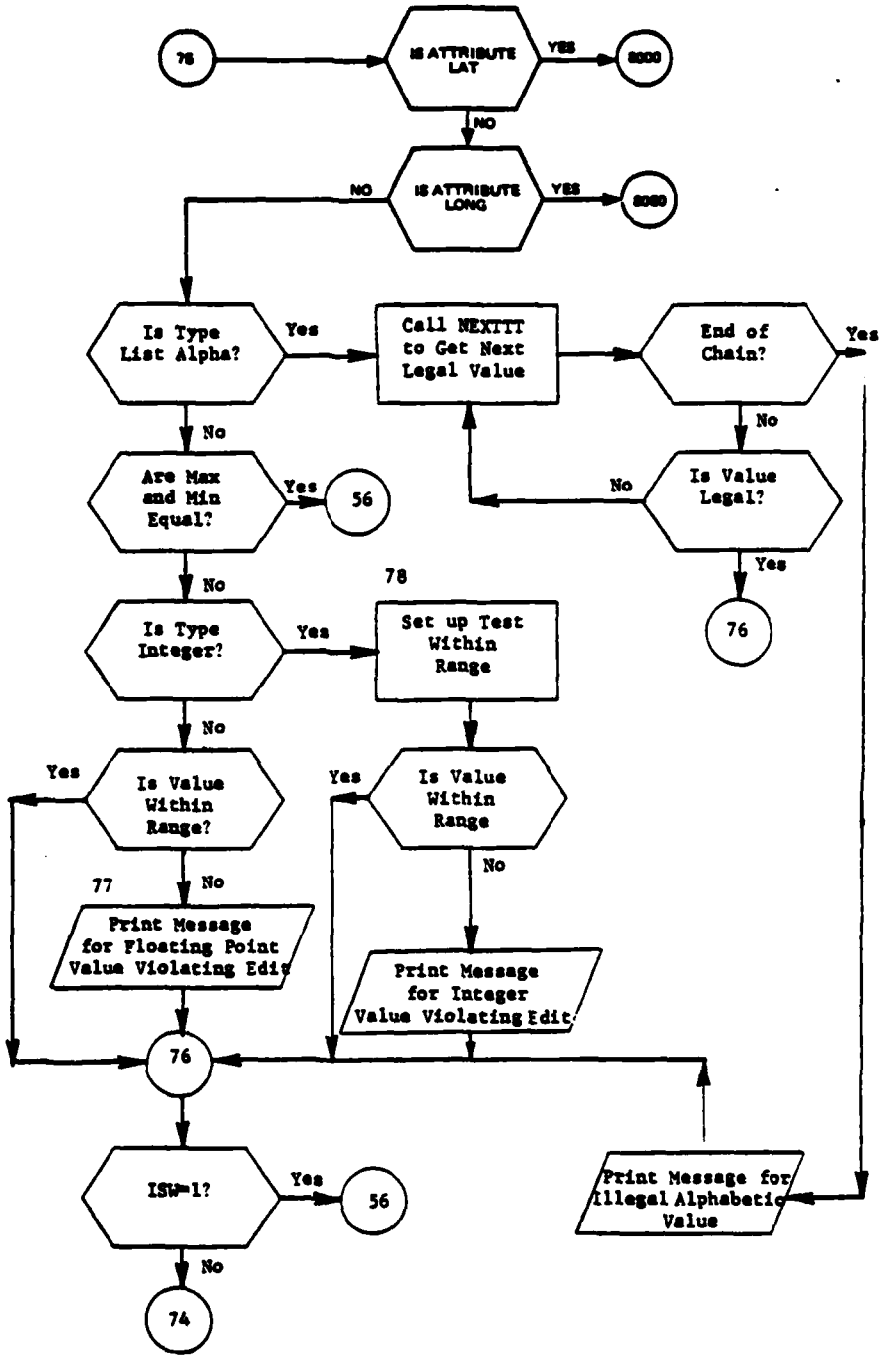


Figure 57. (Part 7 of 8)

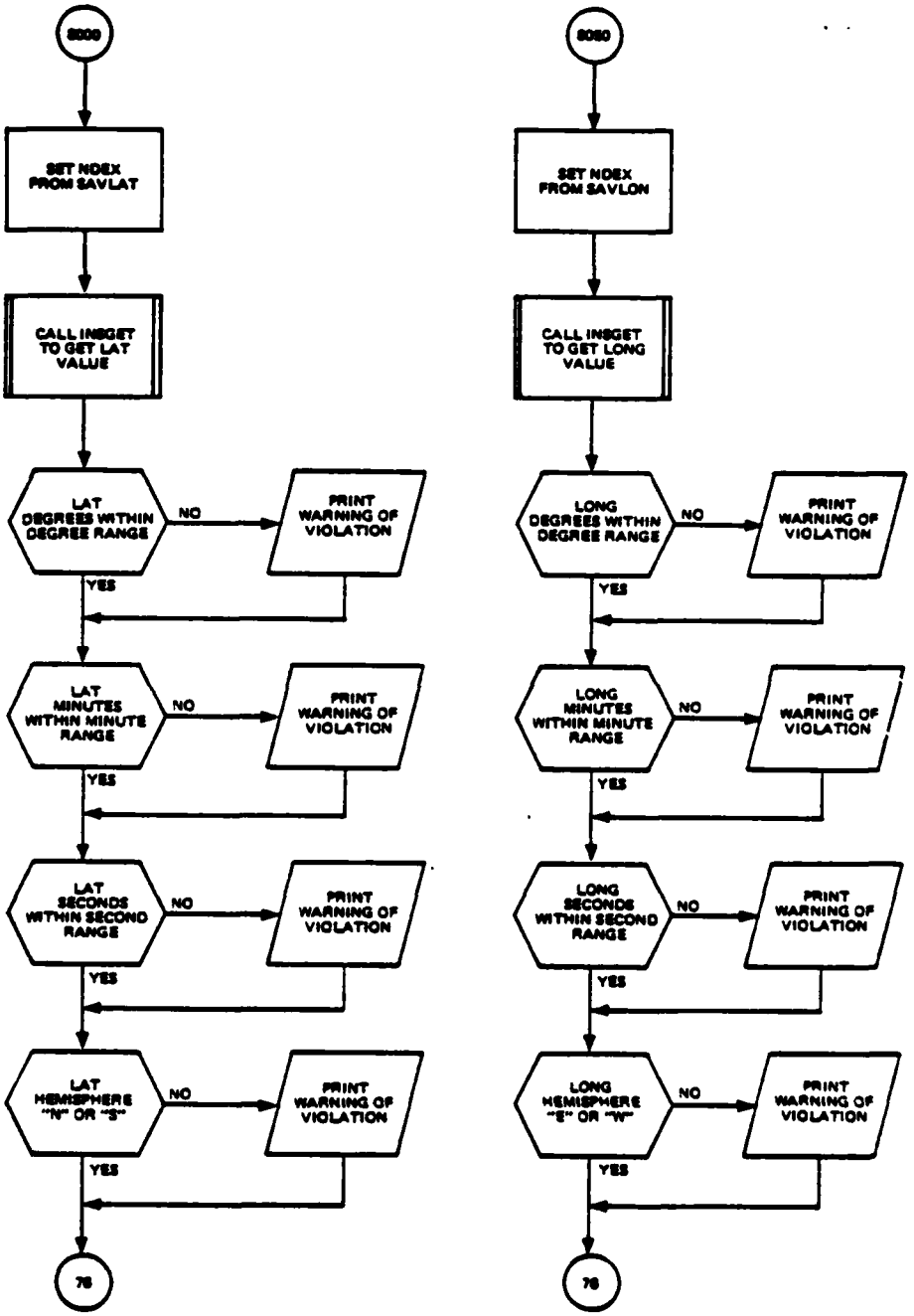


Figure 57. (Part 8 of 8)

THIS PAGE INTENTIONALLY LEFT BLANK

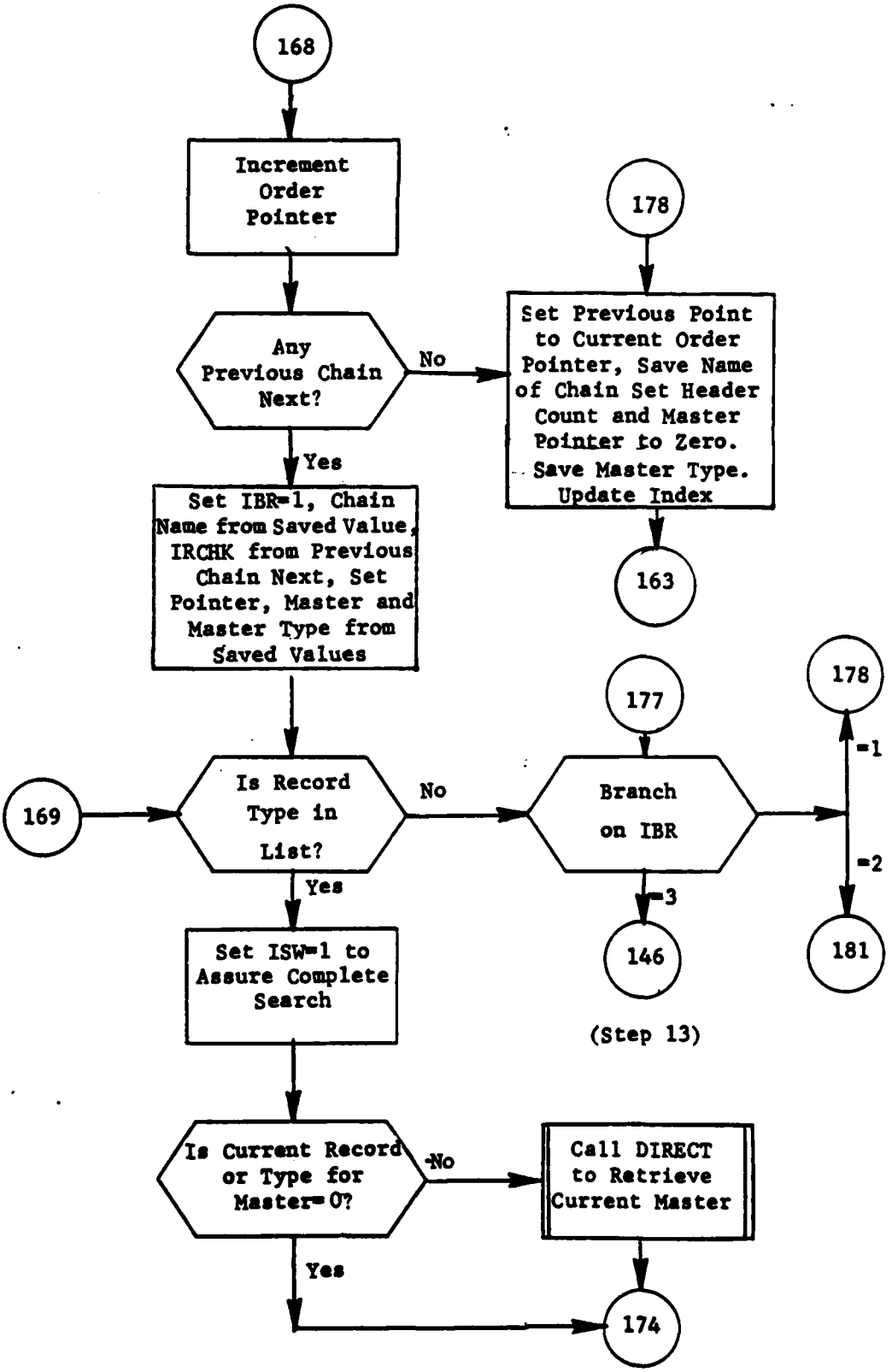


Figure 67. (Part 3 of 6)

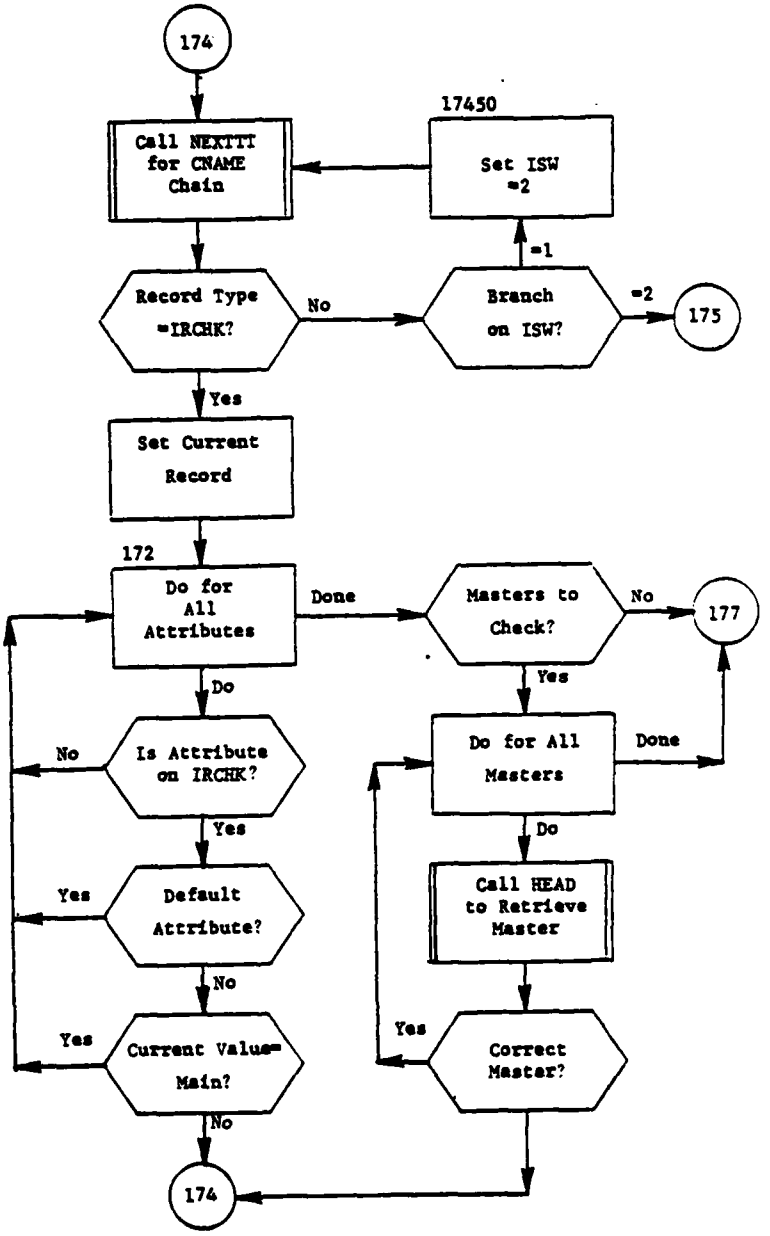


Figure 67. (Part 4 of 6)

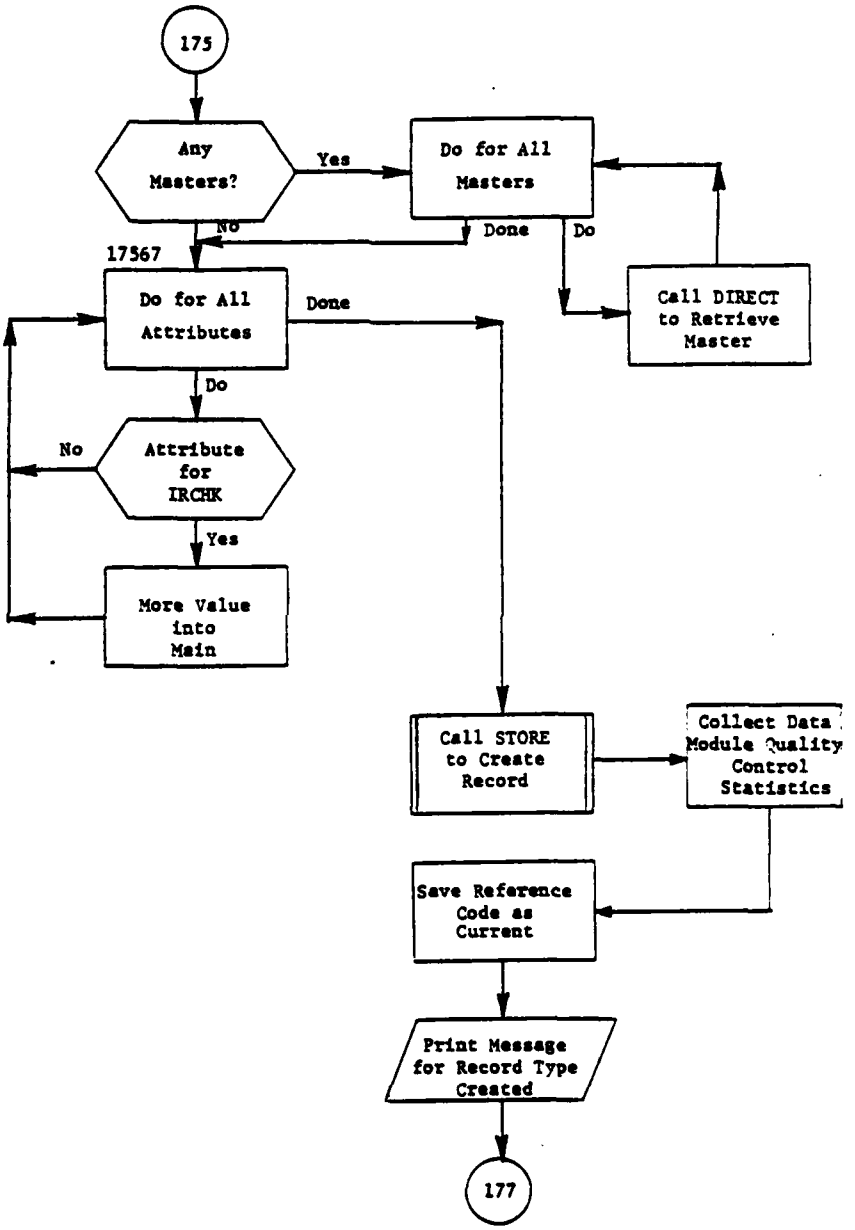


Figure 67. (Part 5 of 6)

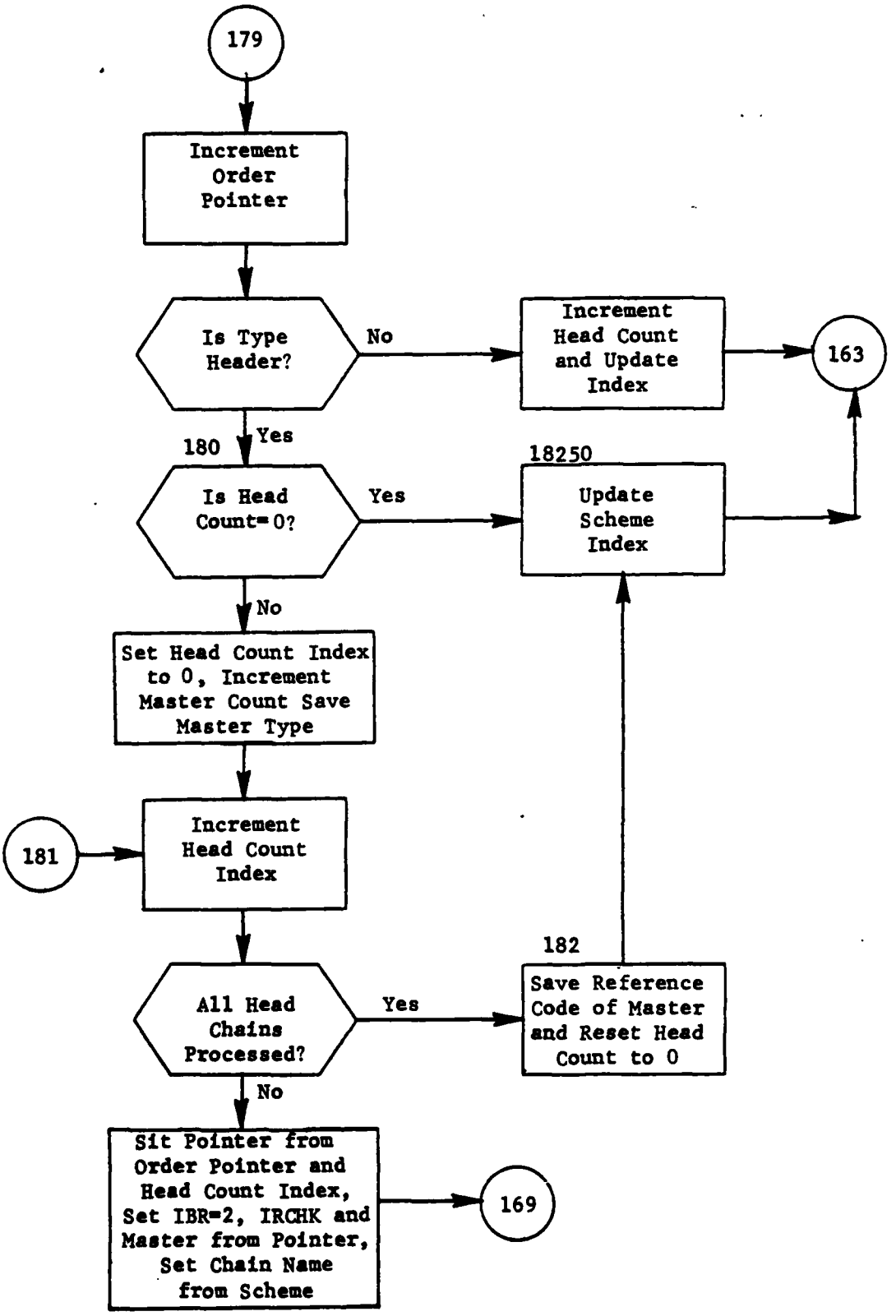


Figure 67. (Part 6 of 6)

4.10 Subroutine DELETE

PURPOSE: To delete records

ENTRY POINTS: DELETE

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C20, C30, OOPS, ORDER, PRWSP, SCHEME, QC

SUBROUTINES CALLED: DLETE, GETNXT, HDFND, HEAD, INSGET, LINKUP, NEXTTT, OFVAL, PRIMHD, SETSCH, UNCODE, VALFND, XWHERE

CALLED BY: ENTMOD (DATA)

Method:

The DELETE verb process can be broken into five steps which are executed in sequence.

Step One

The WHERE clause is scanned. In the process OF phrases and LIKE strings are resolved immediately using VALFND and the values stored by OFVAL. Attributes which are found are placed in a list (ATNUMB) and if either the CLASS or SIDE attributes are encountered the values given are saved to assist in the retrieval scheme construction process.

Step Two

The list of attributes (ATNUMB) is now resolved via the ATRIB chain to create a list of record types (RTLIST). A single defined attribute record type is added to the list. A control attribute has its controlled record added to the list. The record types which contain multiply defined attributes are saved in a separate list (MLTLST).

Step Three

The primary header is now determined in one of two ways. If a value was given for CLASS, HDFND is called for the primary header. Otherwise the highest numbered record is determined and PRIMHD is called. Now the chains of which the primary header is the master are searched looking for matches among the list of multiple attribute record types. Any matches are added to the record type list. Finally LINKUP and SETSCH are called to build the retrieval scheme.

Step Four

The retrieval scheme is used to determine the lowest record type in the hierarchy to be retrieved and this type is noted as the type to be deleted (DNAME).

Step Five

Now GETNXT is called to execute the retrieval scheme. For each record retrieved, XWHERE is called to determine if the record is desired. If so DLETE is called for the DNAME record type.

Subroutine DELETE is illustrated in figure 68,

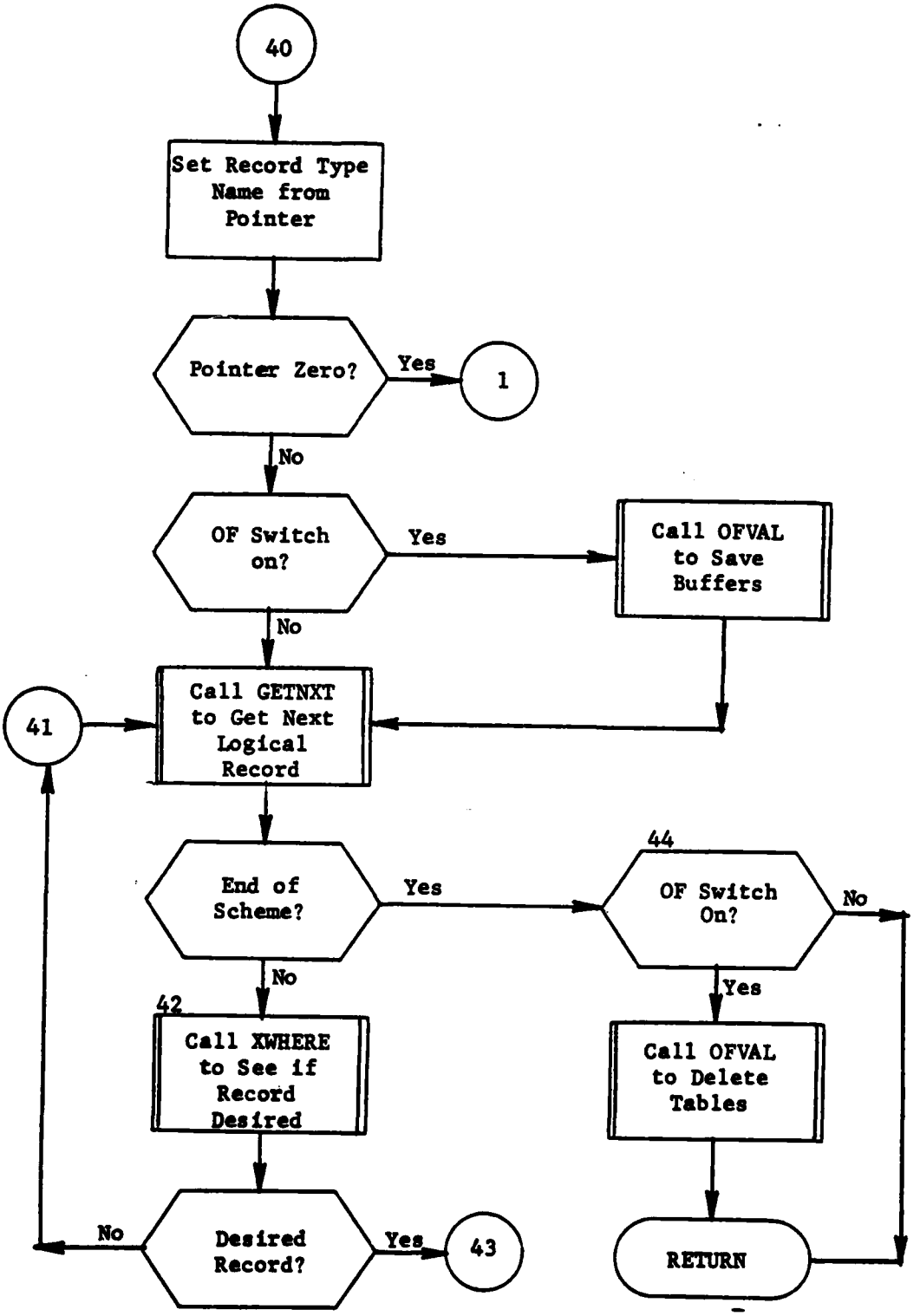


Figure 68. (Part 11 of 12)

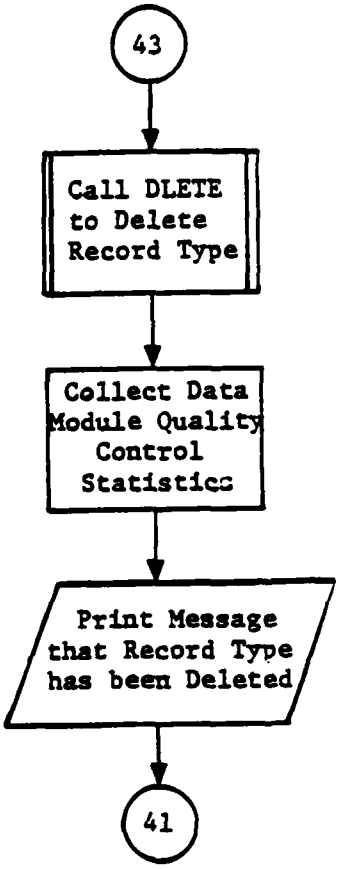


Figure 68. (Part 12 of 12)

DISTRIBUTION

Addressee	Copies
CCTC Codes	
C124 (Reference and Record Set)	3
C124 (Stock)	6
C126	2
C313	1
C314	7
C630	1
DCA Code	
205	1
EXTERNAL	
Chief, Studies, Analysis and Gaming Agency, OJCS ATTN: SFD, Room 1D935, Pentagon, Washington, DC 20301	2
Chief of Naval Operations, ATTN: OP-654C, Room BE781 Pentagon, Washington, DC 20350	2
Commander-in-Chief, North American Air Defense Command ATTN: NPXYA, Ent Air Force Base, CO 80912	2
U.S. Air Force Weapons Laboratory (AFSC) ATTN: AFWL/SUL (Technical Library), Kirtland Air Force Base, NM 87117	1
Director, Strategic Target Planning, ATTN: (JPS), Offutt Air Force Base, NE 68113	2
Defense Technical Information Center, Cameron Station, Alexandria, VA 22314	12
	42

THIS PAGE INTENTIONALLY LEFT BLANK

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (when data entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CS MM 9-77 Volume I, Parts I & II	2. GOVT. ACCESSION NO.	3. RECEIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THE CCTC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK), Program Maintenance Manual, Data Management Subsystem		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR (s) Dale J. Sanders, Paul F. M. Maykrantz, Jim M. Herrin, Edward F. Bersson		8. CONTRACT OR GRANT NUMBER (s) DCA 100-75-C-0019
9. PERFORMING ORGANIZATION NAME & ADDRESS System Sciences, Incorporated 4720 Montgomery Lane Bethesda, Maryland 20014		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME & ADDRESS Command and Control Technical Center Room BE-685, The Pentagon, Washington, DC 20301		12. REPORT DATE 1 June 1977 13. NUMBER OF PAGES 956
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASS./DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this report)

17 DISTRIBUTION STATEMENT (of the abstract entered in block 20, if different from report)

Approved for public release; distribution unlimited.

18. SUPPLEMENTARY NOTES

19. KEY WORDS (continue on reverse side if necessary and identify by block number)

War Gaming, Resource Allocation

20. ABSTRACT (continue on reverse side if necessary and identify by block number)

The computerized Quick-Reacting General War Gaming System (QUICK) will accept input data, automatically generate global strategic nuclear war plans, provide statistical output summaries, and produce input tapes to simulator subsystems external to QUICK.

DD FORM 1 JAN 72 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (when data entered)

20. ABSTRACT (Continued)

The Program Maintenance Manual consists of four volumes which facilitate maintenance of the war gaming system. This volume, Volume I, provides the programmer/analyst with a technical description of the purpose, functions, general procedures, and programming techniques applicable to the modules and subroutines of the Data Management Subsystem.

The Program Maintenance Manual complements the other QUICK Computer Manuals to facilitate application of the war gaming system. These manuals (Series 9-77) are published by the Command and Control Technical Center (CCTC), Defense Communications Agency (DCA), The Pentagon, Washington, DC 20301.

ACKNOWLEDGMENT

This documentation was prepared under the direction of the Chief for Military Studies and Analysis, CCTC, in response to a requirement of the Studies, Analysis, and Gaming Agency, Organization of the Joint Chiefs of Staff. Technical support was provided by System Sciences, Incorporated under Contract Number DCA100-75-C-0019. Change set two was prepared under Contract Number DCA100-78-C-0035. Computer Sciences Corporation prepared change set three under Contract Number DCA100-78-C-0042.

ILLUSTRATIONS (PART II)

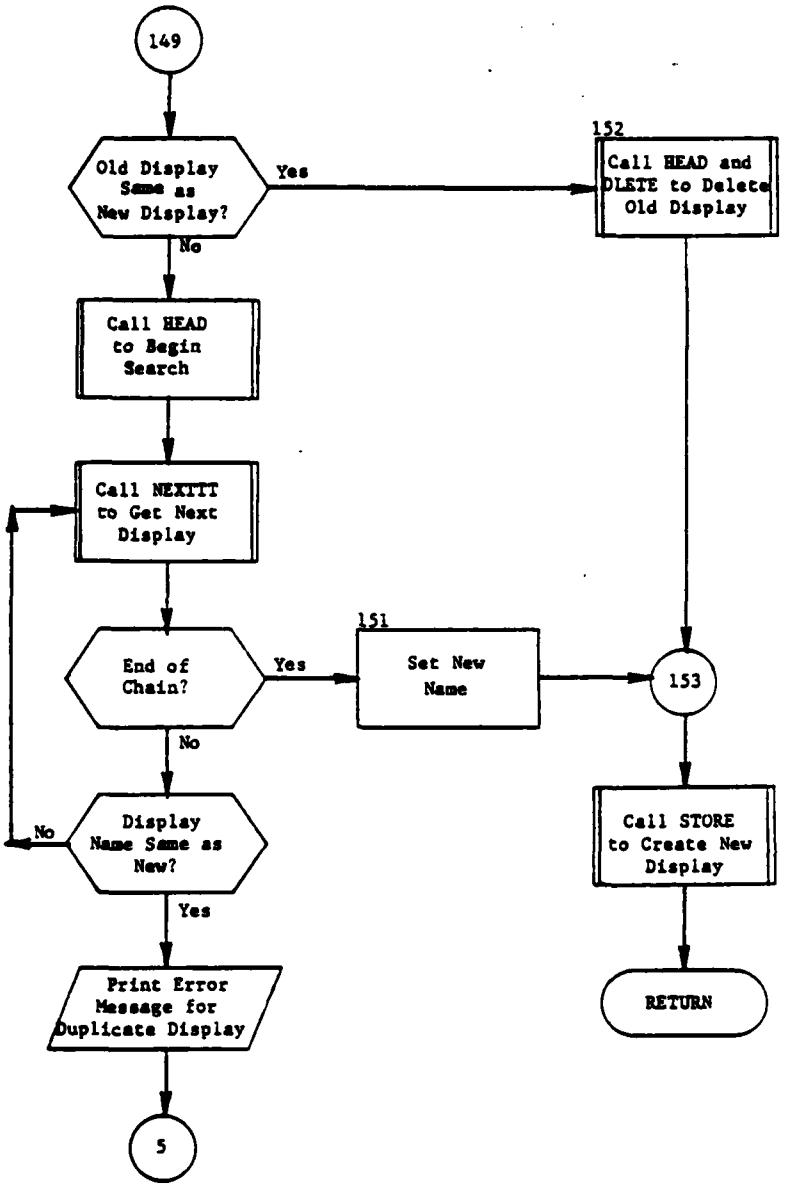


Figure 92. Subroutine ALTER: Step Seven

6.9 Subroutine DESIGN*

PURPOSE: To design a new display

ENTRY POINTS: DESIGN

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C15, C30, DEFNMZ, DSPHED, OOPS, ZEES

SUBROUTINES CALLED: DLETE, HDFND, INSGET, NEXTTT, RETRV, STORE, TABMNT, UNCODE

CALLED BY: ENTMOD (REPORT)

Method:

Step One

First the input is scanned for a DISPLAY clause. If none exists the new display is named 'QTEMPORARYQQ'. If a DISPLAY clause is given, the name is obtained from it. If the name is said to be 'OLD,' the old clause is found and deleted. If the name is said to be 'NEW' a check is made to assure against duplication. The new display table record (DISPRC) is now created (see figure 93).

Step Two

The attributes PAGELENGTH, LINELENGTH, and REPORTCODE are set to their defaults (55, 120 and 42, respectively). Then the input is scanned for a SETTING clause. Any of the attributes named whose values are set by the SETTING clause are altered to reflect the new values (see figure 94).

Step Three

Now any and all DEFINE clauses are read in and stored in TABMNT utility table 1. As DEFINEs are read they are checked for errors and their names are saved in common block DEFNMZ (see figure 95).

Step Four

If there is an input WHERE clause it is read in and stored in utility table 2. As it is read in it is checked for errors (see figure 96).

Step Five

If there is an input SORT clause it is read in and stored in utility table 3 (see figure 97).

*Main routine of overlay RPTDSN

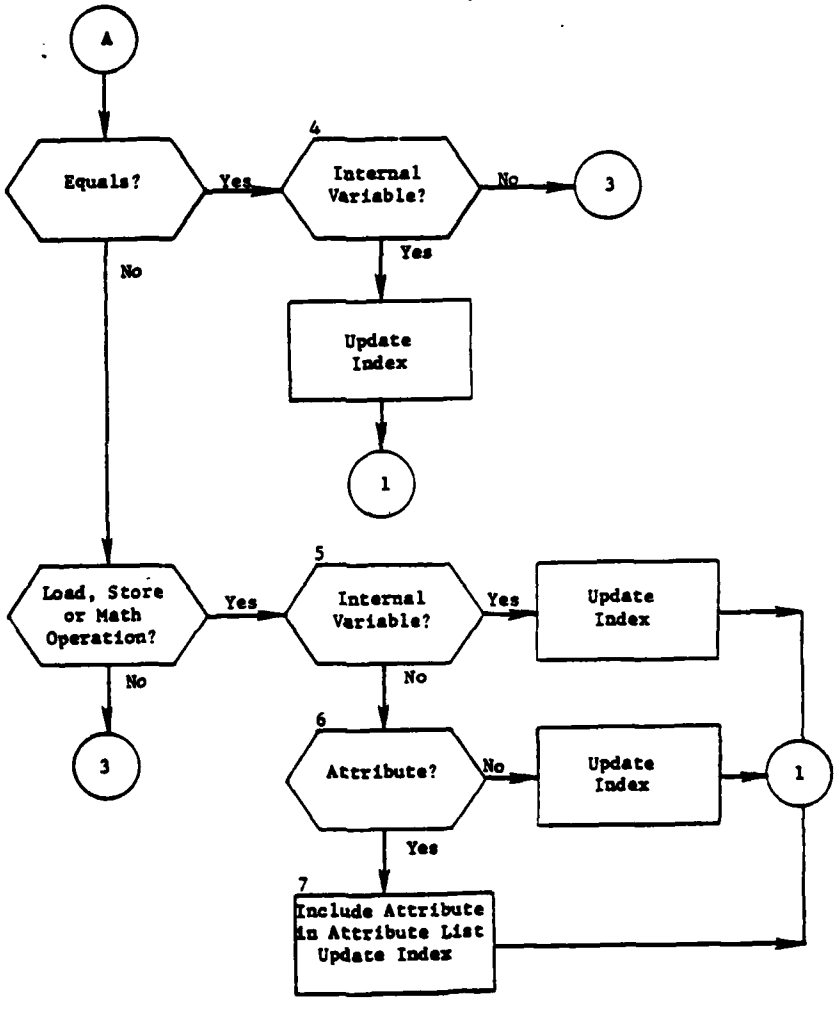


Figure 99. (Part 2 of 2)

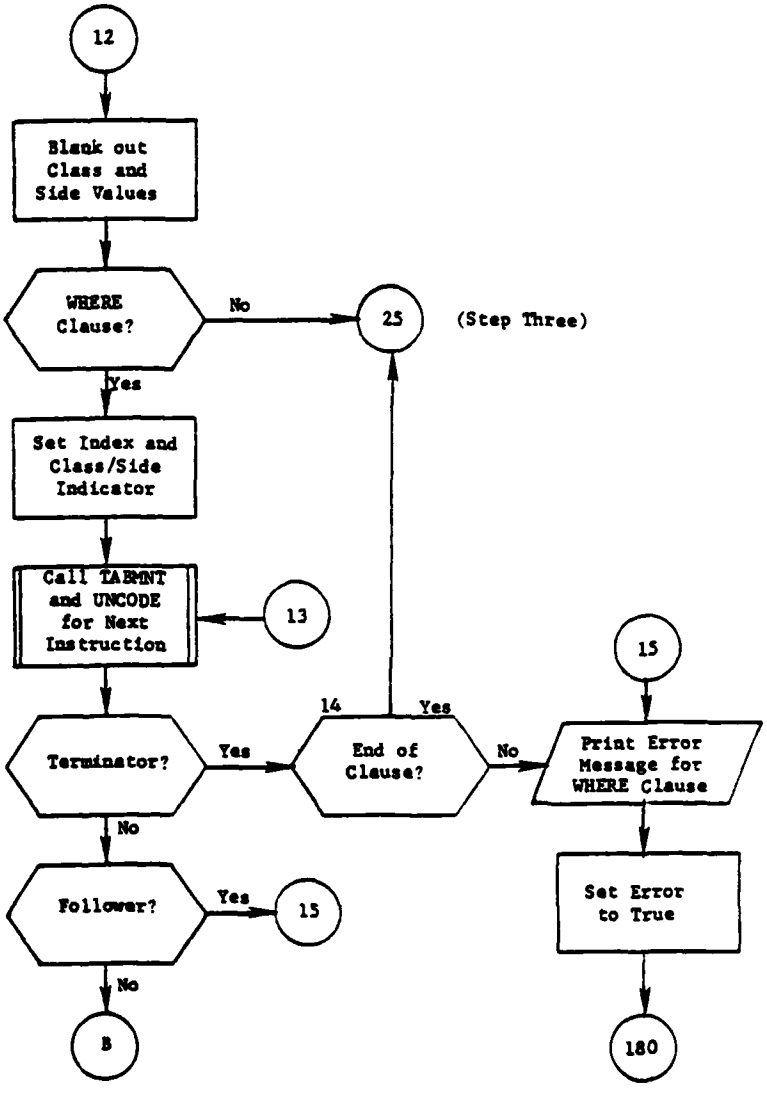


Figure 100. Subroutine DSPMAK: Step Two (Part 1 of 2)

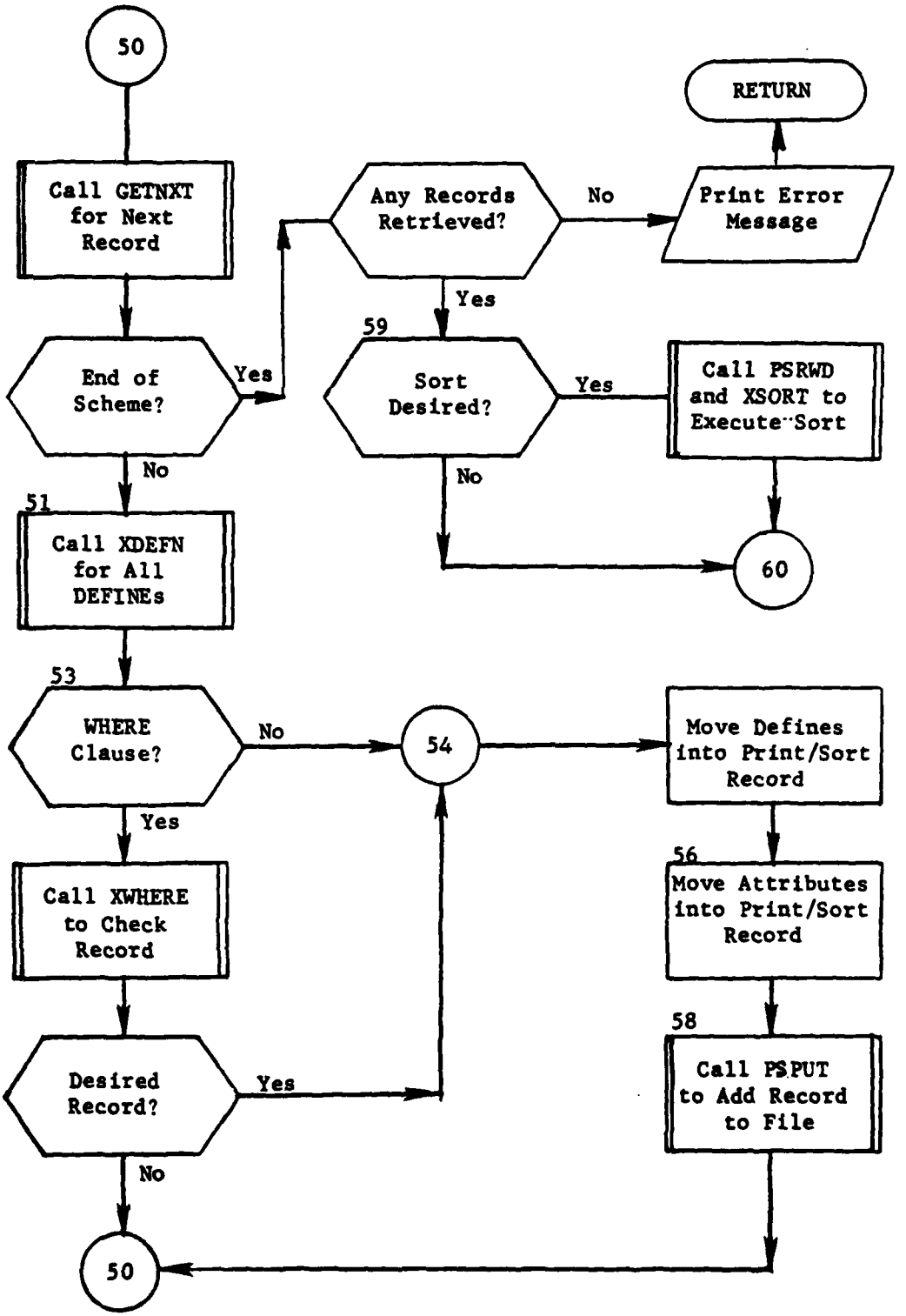


Figure 112. (Part 2 of 2)

CH-1

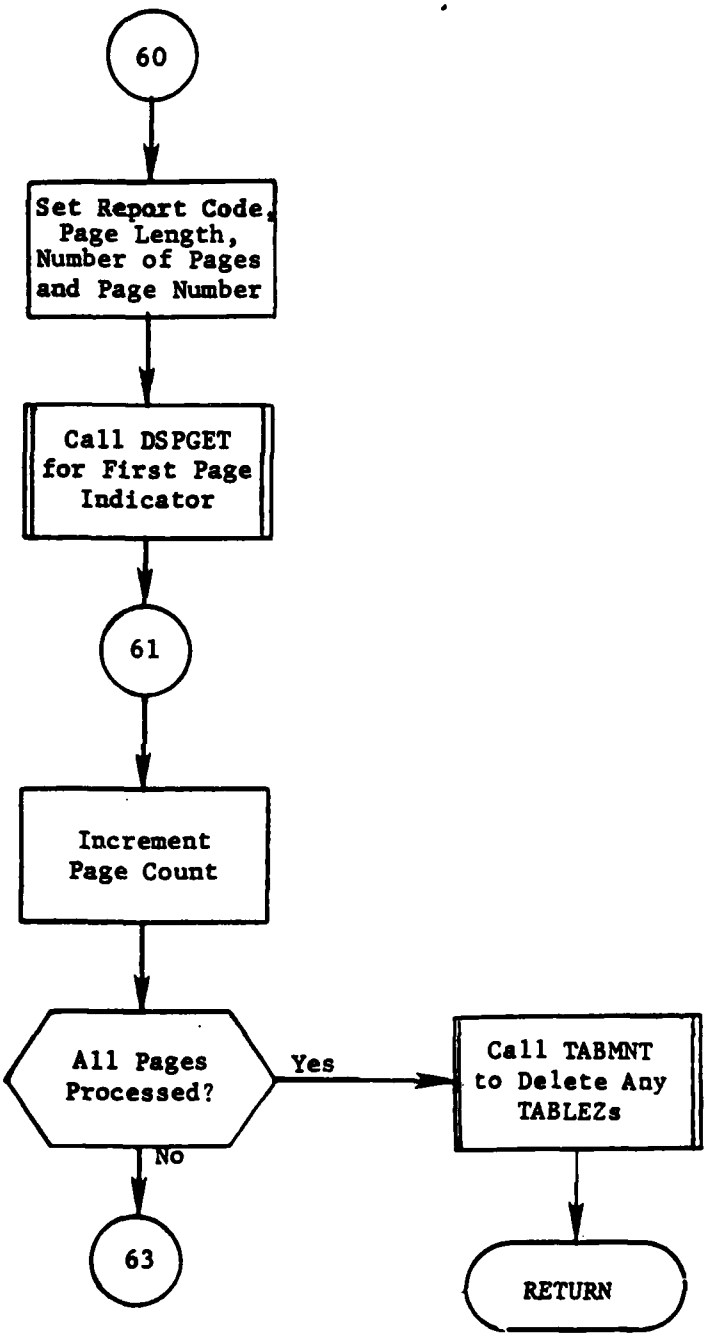


Figure 113. Subroutine PRINCE: Step Five (Part 1 of 8)

CONTENTS OF PAGES 597-598 INTENTIONALLY DELETED

THIS PAGE INTENTIONALLY LEFT BLANK

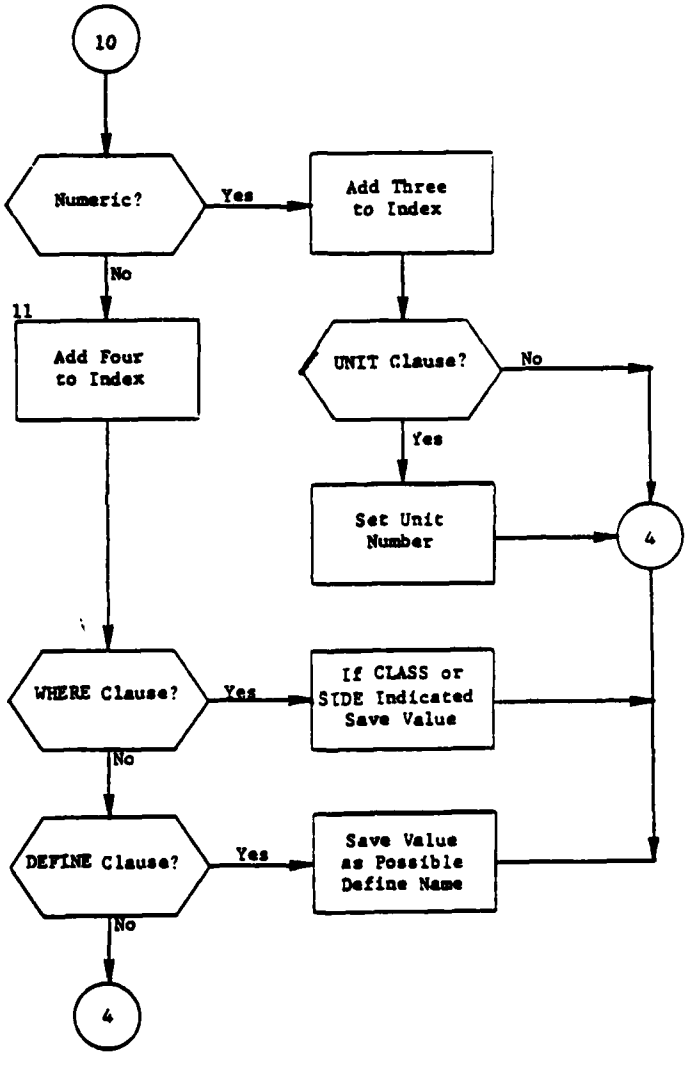


Figure 119. (Part 6 of 7)

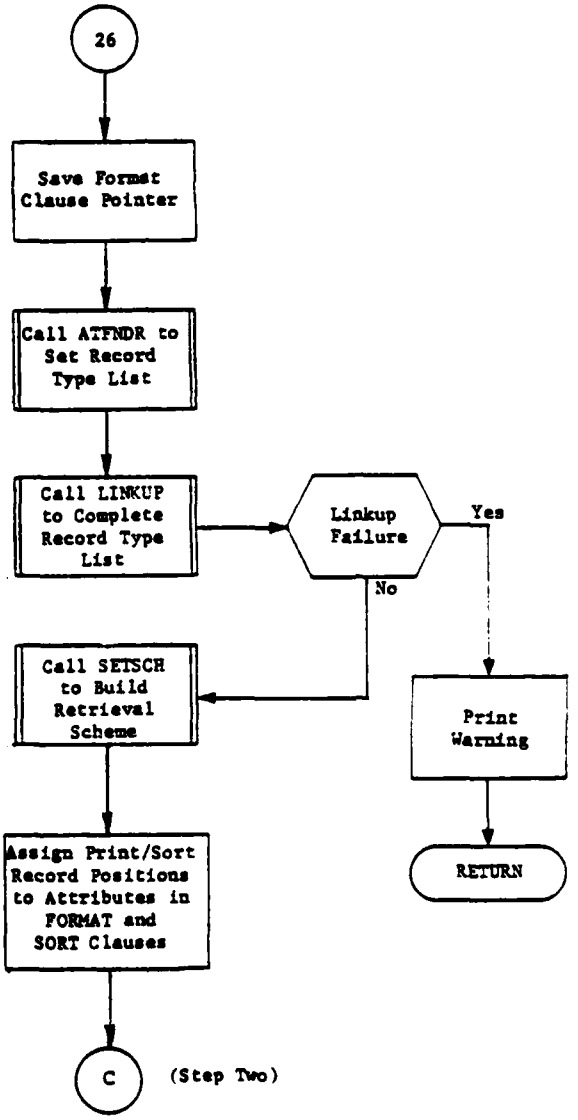


Figure 119. (Part 7 of 7)

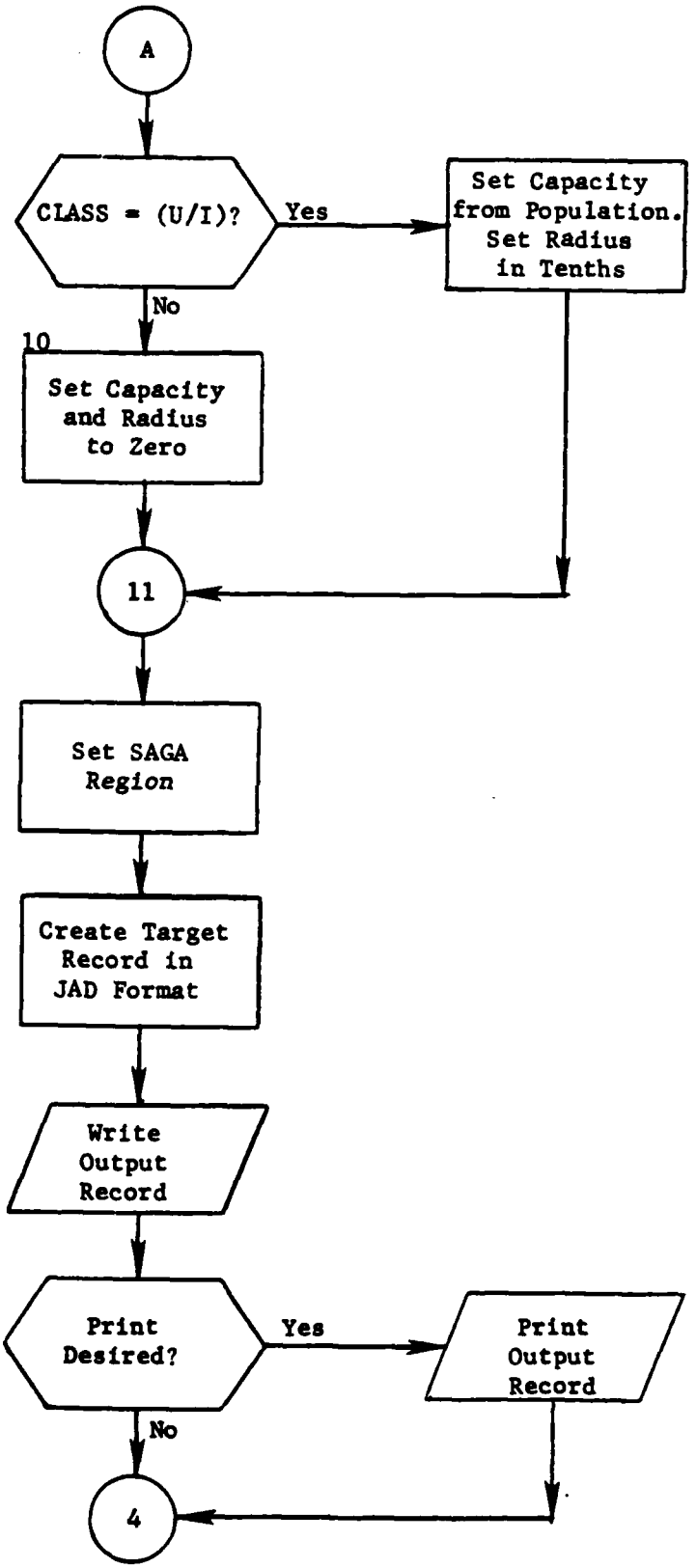


Figure 129. (Part 3 of 3)

8.11 Subroutine TABBLE*

PURPOSE: To produce TABLE output

ENTRY POINTS: TABBLE

FORMAL PARAMETERS: None

COMMON BLOCKS: C10, C20, C30, OOPS, PRINSP, SCHEME

SUBROUTINES CALLED: CONVLL, GETNXT, HEAD, INSGET, NEXTTT

CALLED BY: ENTMOD (EIM)

Method:

The general method is for a preset retrieval scheme to be set in common block SCHEME and executed by GETNXT. Each record retrieved is then written in its particular format. First the input is checked for a UNIT clause which could change the output tape unit (default=35) and for a WHERE clause which could alter the value of the attacking side (default=BLUE).

Next the retrieval scheme to retrieve targets on the defending side of CLASS=MISSIL (array SCHA) is stored, executed and the output written. Then the scheme to retrieve attacking weapon types of CLASS=BMBWEP (SCHB) is stored, executed and the output written. This scheme is then altered to retrieve CLASS=MSLWEP and executed.

Next the retrieval scheme to retrieve attacking warhead types of CLASS=BOMB (array SCHC) is stored and executed. This same scheme is then altered consecutively to retrieve and write out the CLASSs RV, MRV, MIRV and ASM.

Now the retrieval scheme to retrieve the attacking missile bases (array SCHD) is stored, executed and the bases written out. This scheme is altered to retrieve bomber bases and executed. Finally, the scheme for offensive recovery bases (array (SCHE) is used to retrieve and display them.

Subroutine TABBLE is illustrated in figure 130.

*Main routine of overlay BTABLE

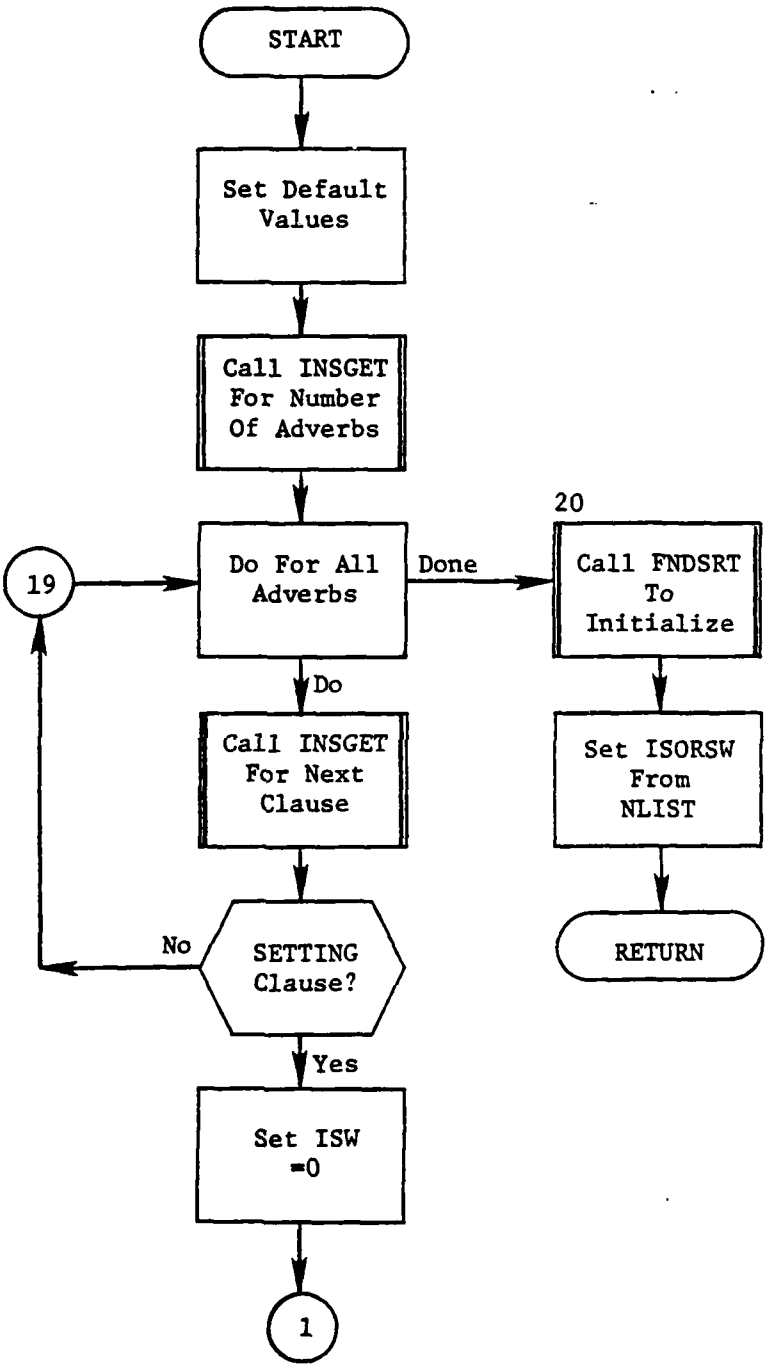


Figure 130.5. Subroutine PLOTINIT (Part 1 of 4)

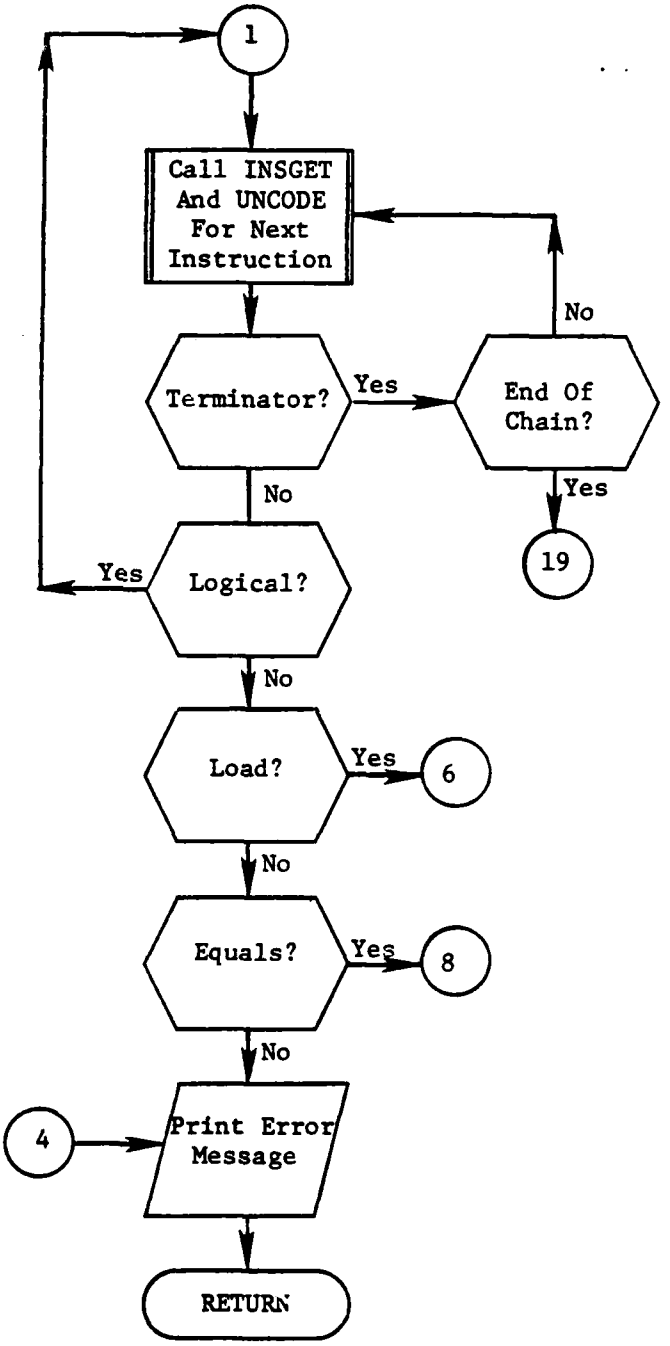


Figure 130.5. (Part 2 of 4)

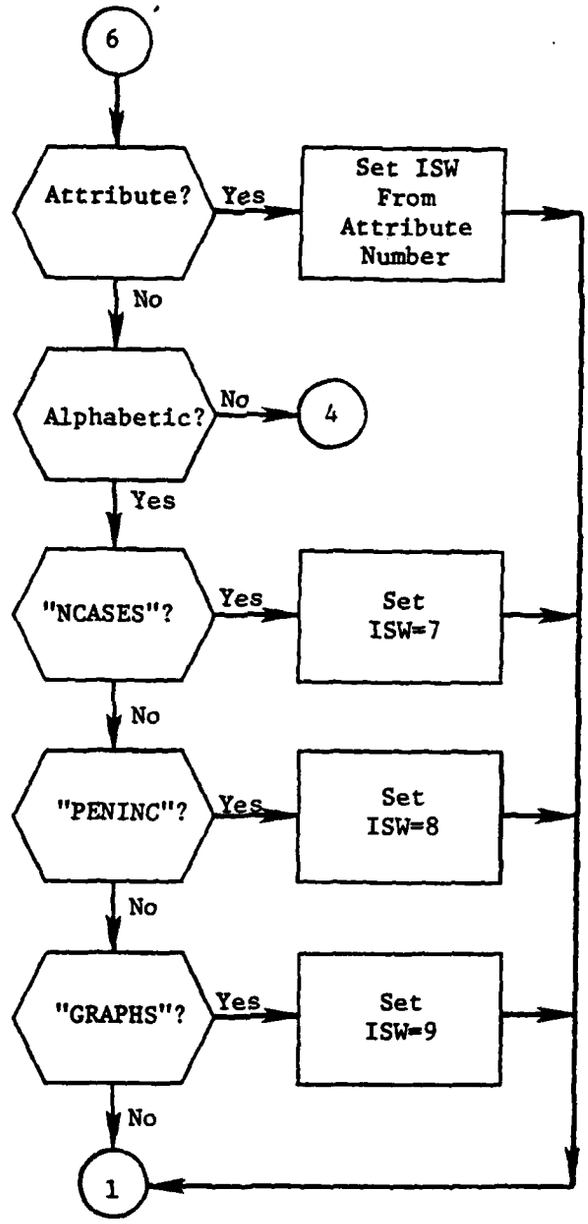


Figure 130.5. (Part 3 of 4)

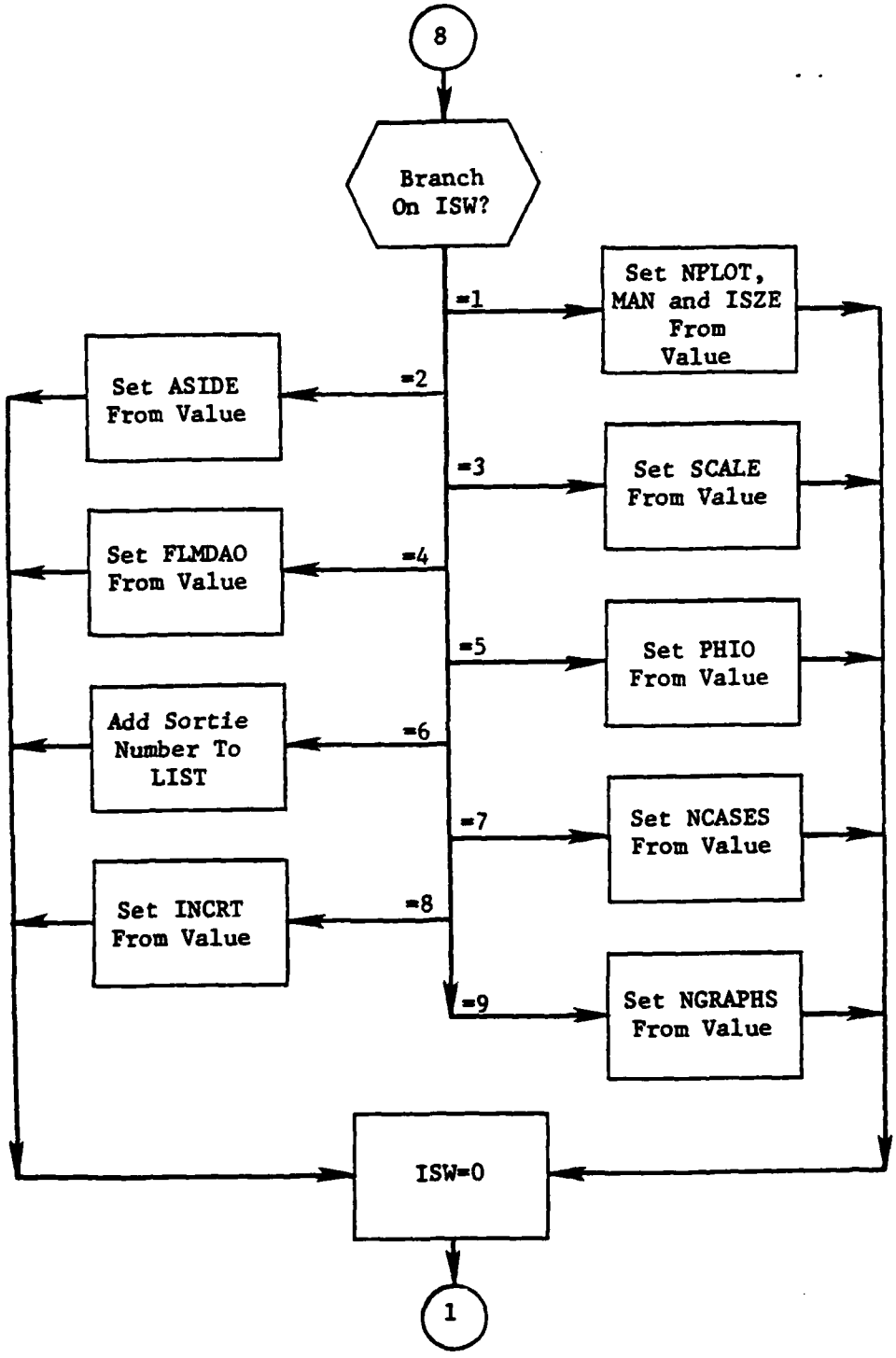


Figure 130.5. (Part 4 of 4)

Table 24. (Part 4 of 5)

Block	Array or Variable	Description
	JHDR	Record type number of primary header
SCHEME	POINT	Pointer to current retrieval scheme instruction
	SCHEME(200)	Retrieval scheme (section 4.4)
SCRTCH	LIST(300)	Storage space used as work area by several subroutines
SIDES	SIDES(5)	List of values for SIDE
SNDMIN	XMIN	Minimum value of x-coordinates
	YATXMN	Y-coordinate at minimum X-coordinate
	XATYMN	X-coordinate at minimum Y-coordinate
	YMIN	Minimum value of y-coordinates
	ISUMIT	Number of points off the graph
SORSCH	SRTSCH(100)	Sort scheme (see section 6.10)
TAPES	PLOTTAPE	Logical unit number for plot tape
	PIECTAPE	Logical unit number for tape for non-plotted points
TARGET	SACB(1,2)	Target coordinates (Latitude and Longitude)
TGTLIM	ITLIM	Number of pairs of targets
	DESLIM	(2,30) First two character of DESIGs to be excluded
TSTUFF	TOFMIN	Minimum time of flight
	CMISS	Missile time coefficient
	RNGMIN	Minimum range
	RANGEM	Maximum range

Table 24. (Part 5 of 5)

Block	Array or Variable	Description
WAROUT	IWARFL	Logical unit number for printed output
XMEDGE	YMEDGE	Map edge
	XLL	X-coordinate of last point
	YLL	Y-coordinate of last point
	XL	X-coordinate of point to be plotted
	YL	Y-coordinate of point to be plotted
	XWEDGE	Converted value for latitude of origin
	BANGL	Converted value for longitude of origin

9.6.1 Subroutine CONVLL

PURPOSE: Convert latitude and longitude to degrees, minutes, seconds (DMS) format

ENTRY POINTS: CONVLL

FORMAL PARAMETERS:

XLAT:	Input latitude
XLONG:	Input longitude
CHLAT:	Output latitude (character *7)
CHLONG:	Output longitude (character *8)

COMMON BLOCKS: None

SUBROUTINES CALLED: None

Method:

The process is similar for both latitude and longitude. The latitude is converted first. The letter (CHM) is set to N and if the latitude is negative it is set to positive and CHM is set to S. The degrees, minutes and seconds are then broken out and ENCODEd into CHLAT. Longitude is now processed, CHM is set to W. If longitude is greater than 180 it is subtracted from 360 and CHM is set to E. Longitude is then broken down and ENCODED into CHLONG.

Subroutine CONVLL is illustrated in figure 118.

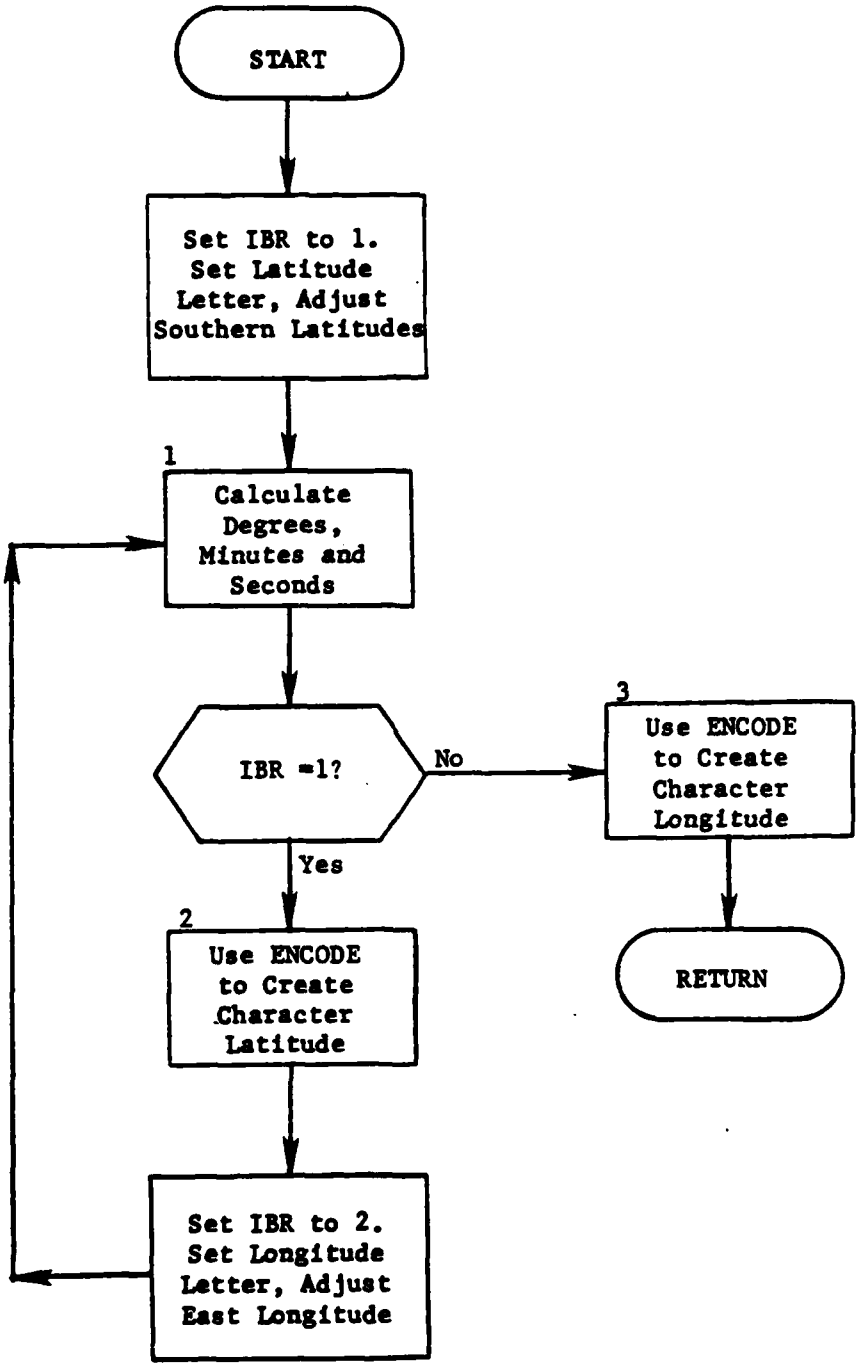


Figure 135.1 Subroutine CONVLL

9.25 Subroutine IORFL

PURPOSE: To check input value and convert it to floating point if it is integer

ENTRY POINTS: IORFL

FORMAL PARAMETERS: X: Value to be checked and, if necessary, converted

COMMON BLOCKS: None

SUBROUTINES CALLED: None

Method:

First the input value is checked to see if it is a floating point zero. If so the routine exits. If not, bits 8 and 9 of the value are compared. If they are equal the value is integer and is converted to floating point.

Subroutine IORFL is illustrated in figure 155.

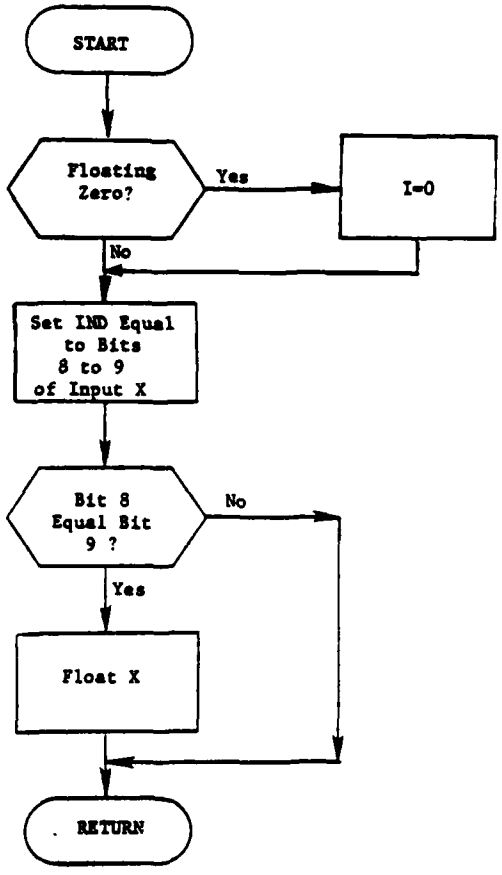


Figure 155. Subroutine IORFL

9.32 Subroutine LINKUP

PURPOSE: To add to a list of record types any additional types needed to build a retrieval scheme

ENTRY POINTS: LINKUP

FORMAL PARAMETERS:
- LIST: List of record type numbers
- LISTLN: Length of list
- KHDR: Record type number of primary header

COMMON BLOCKS: C10, C20, C30, SCRTCH

SUBROUTINES CALLED: NEXTTT

Method:

This subroutine is best understood by examining figure 162. Basically, the routine begins with the input LIST and pairs to it the JIST array which contains an identifying number for the 'complex' of record types to which each record type in LIST belongs. A complex (as defined here) is a group of record types which is continuous in that every record type in the group is linked to every other type in the group through one or more data base chains. The objective of the LINKUP process is to add record types to LIST in such a way that all record types end up in the same complex. If an error condition occurs the length of the output LIST is set to zero (LISTLN=0).

In the first part of the process JIST is set to 0 for all but the primary header (KHDR) which is set to 1. Then each element of LIST is examined as follows. If JIST is zero, it is set to the next number in sequence (ICOMP). Then for every chain of which it is master the detail record is compared to LIST. If a match is found and the match entry has JIST=0, JIST is set equal to JIST for the current LIST element. If a match is found and the match entry has JIST≠0 then all JIST entries equal either to the match entry for JIST or the current entry for JIST are set equal to the lower of these two.

If no match is found for the detail record type in LIST, it is compared to the auxiliary table KIST. If a match is found in KIST, the KIST entry is removed and added to LIST. Furthermore, all entries equal to the JIST current value or the auxiliary table complex number (MIST) for the KIST entry are set equal to the lower numbered value. This reset is done in both JIST and MIST.

If no match is found for the detail record in either LIST or KIST, the detail is added to KIST and the appropriate value set in MIST. When all entries of LIST have been processed, JIST is checked to see if any entries are not equal to 1. If any are not, a more complex process must

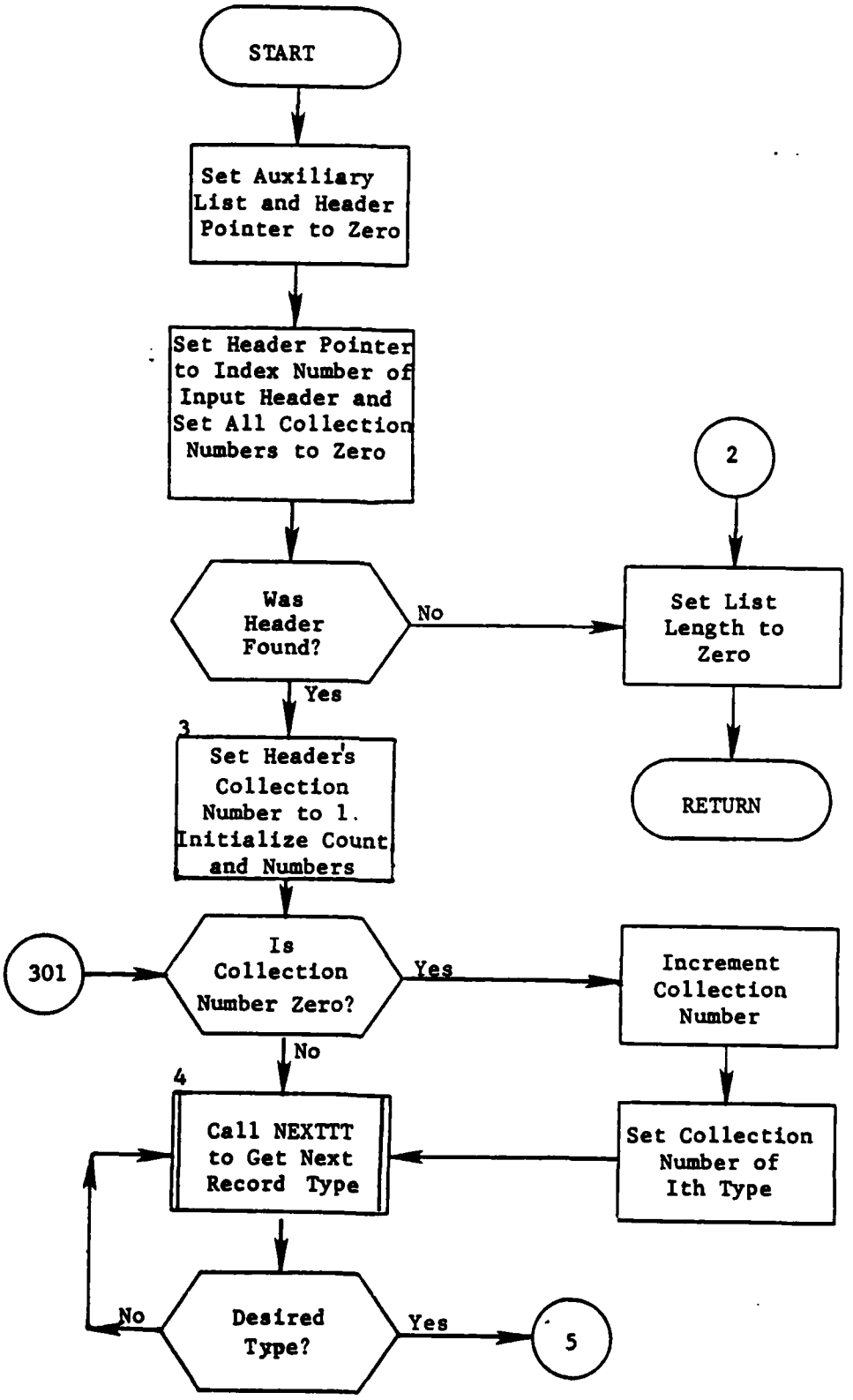


Figure 162. Subroutine LINKUP (Part 1 of 8)

9.42.1 Subroutine PROCTIM

PURPOSE: Obtains remaining processor time from Honeywell system.

ENTRY POINT: TIMOUT

FORMAL PARAMETER: LTIM

COMMON BLOCKS: None

SUBROUTINES CALLED: GEINFO

Method:

Calling parameters to perform a MME GEINFO call are set. The call is then made and the remaining processing time is loaded into LTIM.

Subroutine PROCTIM is illustrated in figure 172.1.

Figure 172.1. Subroutine PROCTIM

THIS PAGE INTENTIONALLY LEFT BLANK

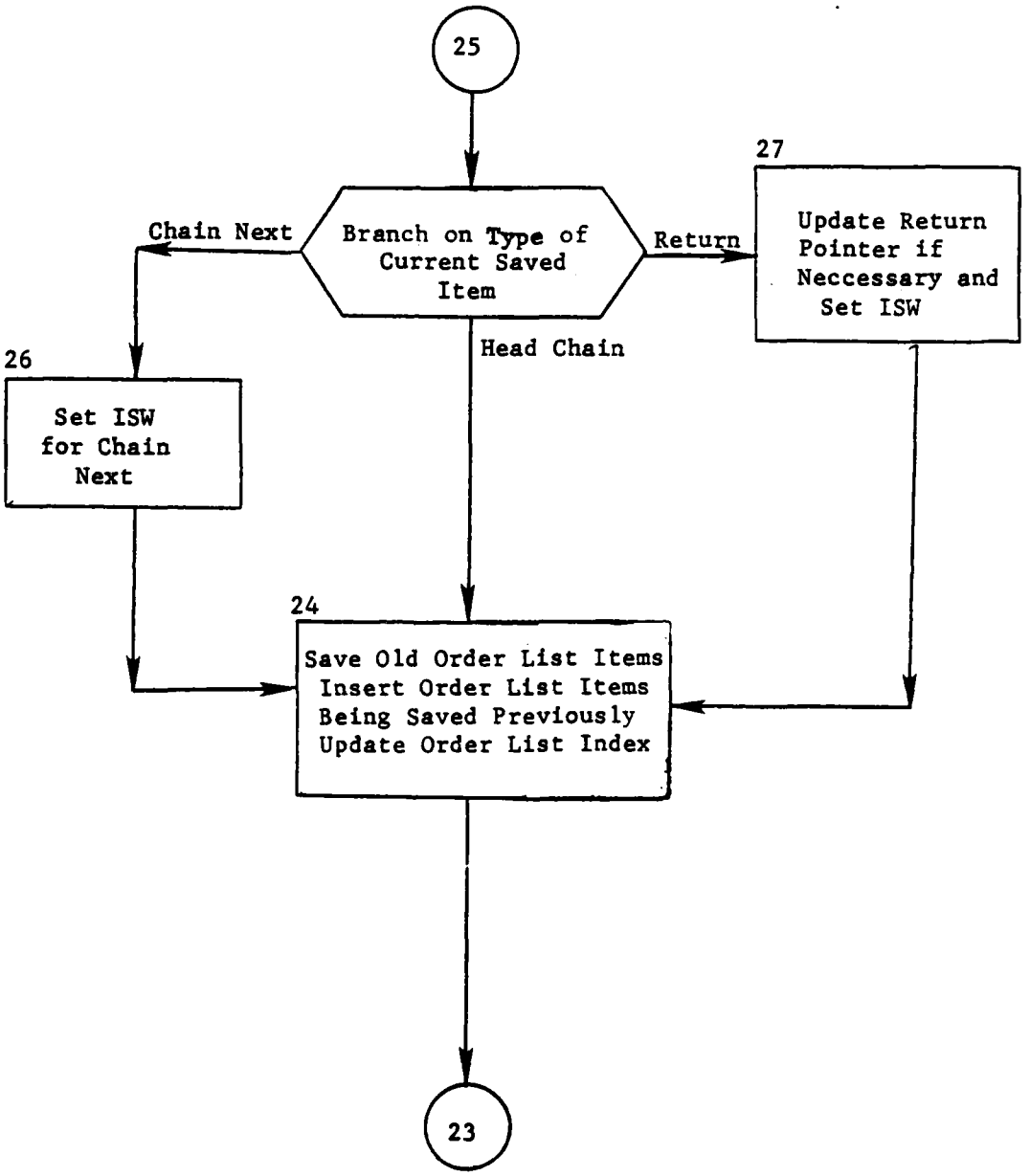


Figure 177. (Part 8 of 8)

9.48 Function SLOG

PURPOSE: To pack QUICK system logical array values (0 = False; 1 = True).

ENTRY POINT: SLOG

FORMAL PARAMETERS:

L	- QUICK logical array
E	- Expression giving value (True is non-zero)
I,J,K	- Array indices
LL,M,N	- Array dimensions

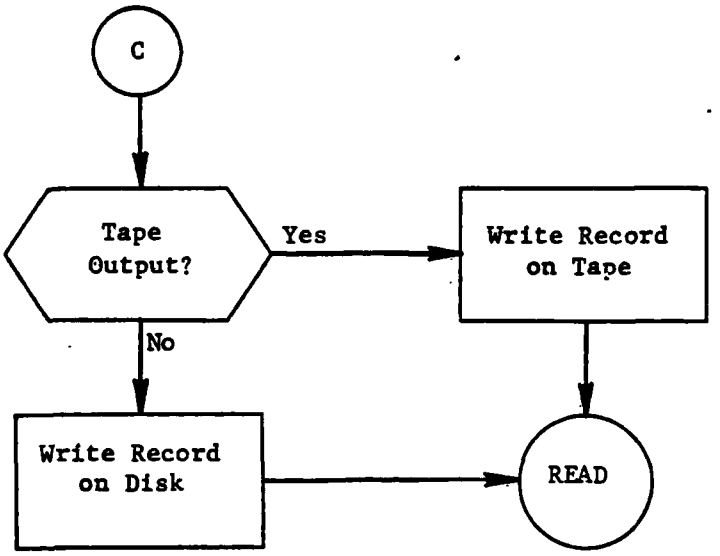
COMMON BLOCKS: None

SUBROUTINES CALLED: None

Method:

The bit position of the desired logical value is determined; the bit is set to zero (.FALSE.) if E equals zero, or to one (.TRUE.) if E is non-zero.

Function SLOG is illustrated in figure 178.



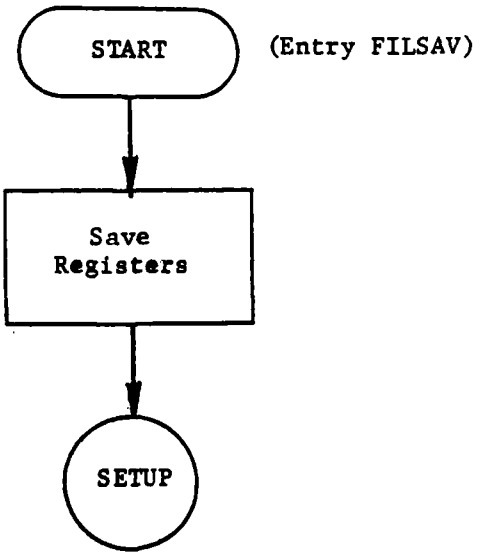


Figure 182. Subroutine SVTP: Entry FILSAV (Part 5 of 5)

9.53 Subroutine SWTCHT

PURPOSE: To set switches corresponding to bit positions 18 thru 35 on the GCOS SWITCH word. These bits are set using the $SET card in the JCL stream.

ENTRY POINTS: SWTCHT

FORMAL PARAMETERS: I - Bit location to be tested
J - Value returned (1 is off, 2 is on)

COMMON BLOCKS: None

SUBROUTINES CALLED: GESETS

Method:

The switch word is obtained from the GCOS system by calling GESETS. The appropriate bit is checked and the return values are set.

Subroutine SWTCHT is illustrated in figure 183.

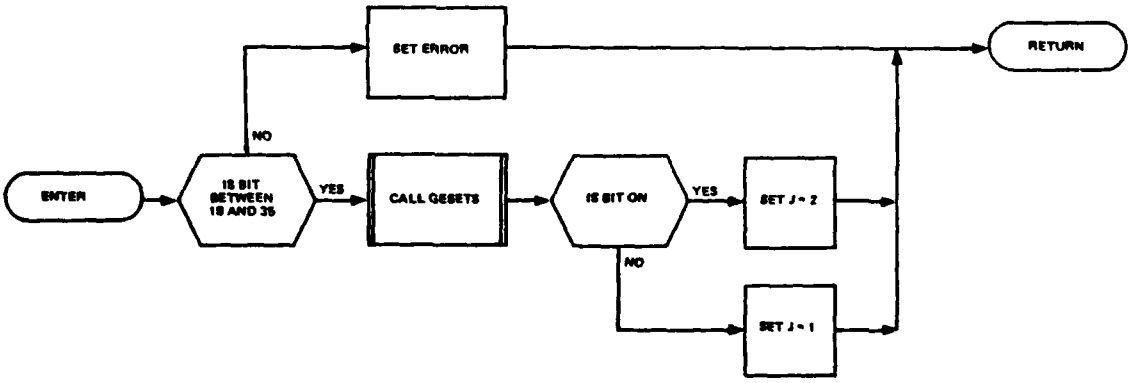


Figure 183. Subroutine SWTCHT

CONTENTS OF PAGES 849-851 INTENTIONALLY DELETED

9.54 Subroutine TGTLIM

PURPOSE: To provide a source for the user-selected targets which are not to be considered by MIRVDUMP.

ENTRY POINTS: TGTLIM

FORMAL PARAMETERS: None

COMMON BLOCKS: TGTLM

SUBROUTINE CALLED: None

Method:

The number of target sets to be excluded from consideration (ITLIM) and pairs of excluded DESIGs ((DESLIM(1,J) and DESLIM(2,J)) where J ranges from 1 to ITLIM are stored in the TGTLM common block. The DESIG identifiers are the first two characters of the DESIG. All targets whose DESIGs are within these bounds will be excluded.

During compilation, the print of this subroutine is suppressed. In this manner, the total compilation print of the UTILITY module can remain unclassified even if the common block TGTLM contains classified data.

There are no executable statements so no flow diagram is shown.

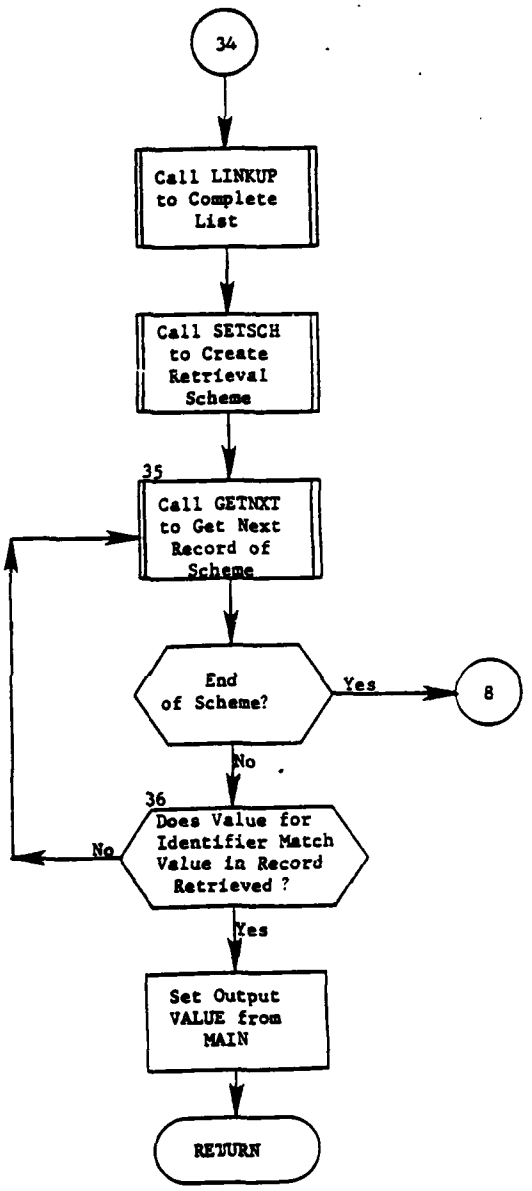


Figure 188. (Part 8 of 8)

9.59 Function VALTAR

PURPOSE: To compute the fraction of target value at weapon time of arrival.

ENTRY POINTS: VALTAR

FORMAL PARAMETERS:

- FV(I) - fraction of value available at the Ith component
- TV(I) - time of the Ith component
- NV - number of value - time components (1-5)
- M - indicator of functional form of time curve
  - = 1 use step-linear form
  - = 2 use exponential form (not presently implemented)
- T - time for which value is desired

COMMON BLOCKS: None

SUBROUTINES CALLED: None

CALLED BY: RECON, EVALPLAN, PROCCOMP, CALCOMP, SALVAL

Method:

VALTAR uses a linear interpolation formula to compute the fraction of target value remaining at weapon time of arrival (parameter T), where FV and TV are the arrays of fractional value remaining and time forming a step-linear function of fraction of target value remaining versus time.

The logic of function VALTAR is shown in figure 189.

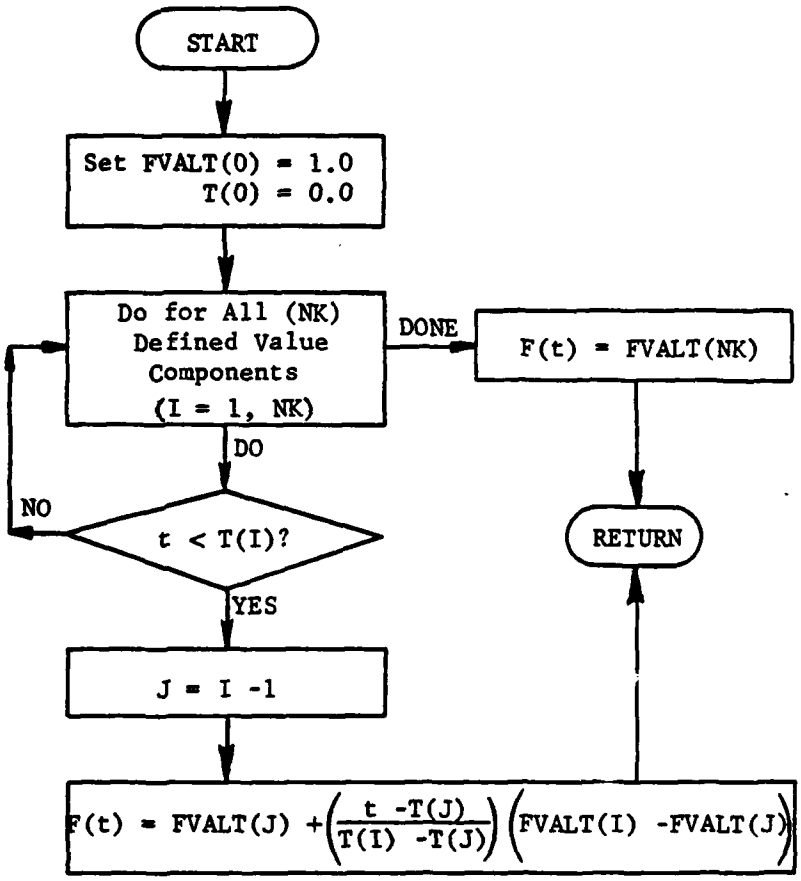


Figure 189. Function VALTAR

9.59.1 Function VLRADP

PURPOSE:
1. Find lethal radius of weapon
2. Set FN for use by calling subroutine

ENTRY POINTS: VLRADA, VLRADP

FORMAL PARAMETERS:
YIELD - Yield of weapon in megatons
NVN - Vulnerability parameter of target
HOB - Weapon height of burst
FN - Parameter specifying shape of damage function

COMMON BLOCKS: DPOOL, PLSTCL

SUBROUTINES CALLED: None

Method:

Entry VLRADA is executed if called from subroutine CALCOMP (local parameter IN is set to nonzero); else VLRADP is executed (local parameter is set to zero). VLRADP entry implies air burst lethal radius is to be calculated purely on target vulnerability; VLRADA entry implies air burst lethal radius is to be calculated based on target vulnerability and scaled height of burst. Ground burst lethal will be calculated the same for both entry points.

NVN is decoded into the appropriate vulnerability number VN, the latter (P or Q), and K-factor XK. The cube root of the yield is extracted. Then the adjusted vulnerability number AVN is determined by methods described in "Computer Computation of Weapon Radius," B-139-61, Air Force Intelligence Center. FN is set to six or three for P and Q type targets, respectively.

The natural logarithm of the lethal radius (in nautical miles) of a 1-megaton burst is contained in arrays PG, QG, QA, PA, QQA, and PPA. Function VLRADP interpolates in the appropriate array to find the logarithm of the 1-megaton lethal radius for AVN. Arrays PG, QG, QA, and PA are at intervals of five vulnerability numbers. The first index of arrays QQA and PPA are also at intervals of hundreds of feet for an air burst. The lethal radius of the weapon is then determined by exponentiating and multiplying by the cube root of the yield.

A flowchart for VLRADP is shown in figure 189.1

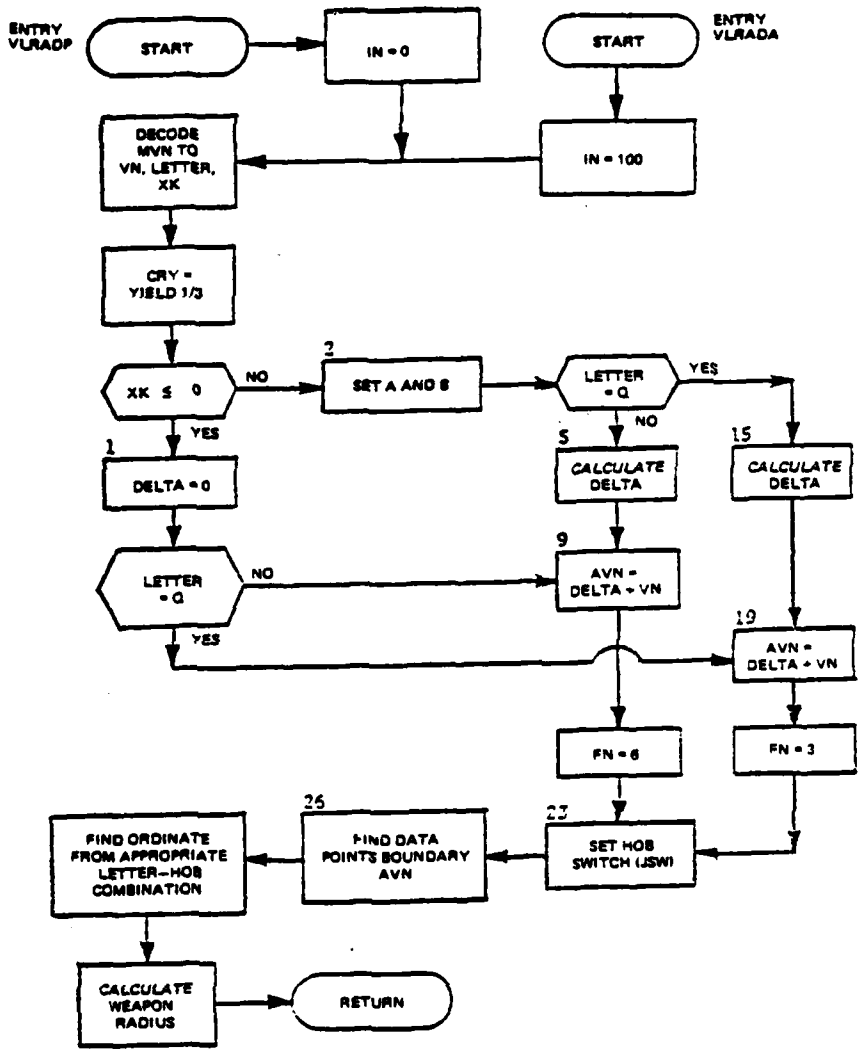


Figure 189.1. Function VLRADP

9.60 Function XLL

PURPOSE: To convert latitude or longitude from DDMMSS format to decimal degrees.

ENTRY POINTS: XLL

FORMAL PARAMETERS: CHRIN - Input character string

COMMON BLOCKS: None

SUBROUTINES CALLED: ABORT

Method:

The input string -- CHRIN -- is scanned one character at a time until one of the characters "N", "S", "E", or "W" is found. Each time a number is found, it is added to 10 times the previous total. When the directional letter is found, the calculated total is converted to decimal degrees and signed based on the directional letter.

Function XLL is illustrated in figure 190.

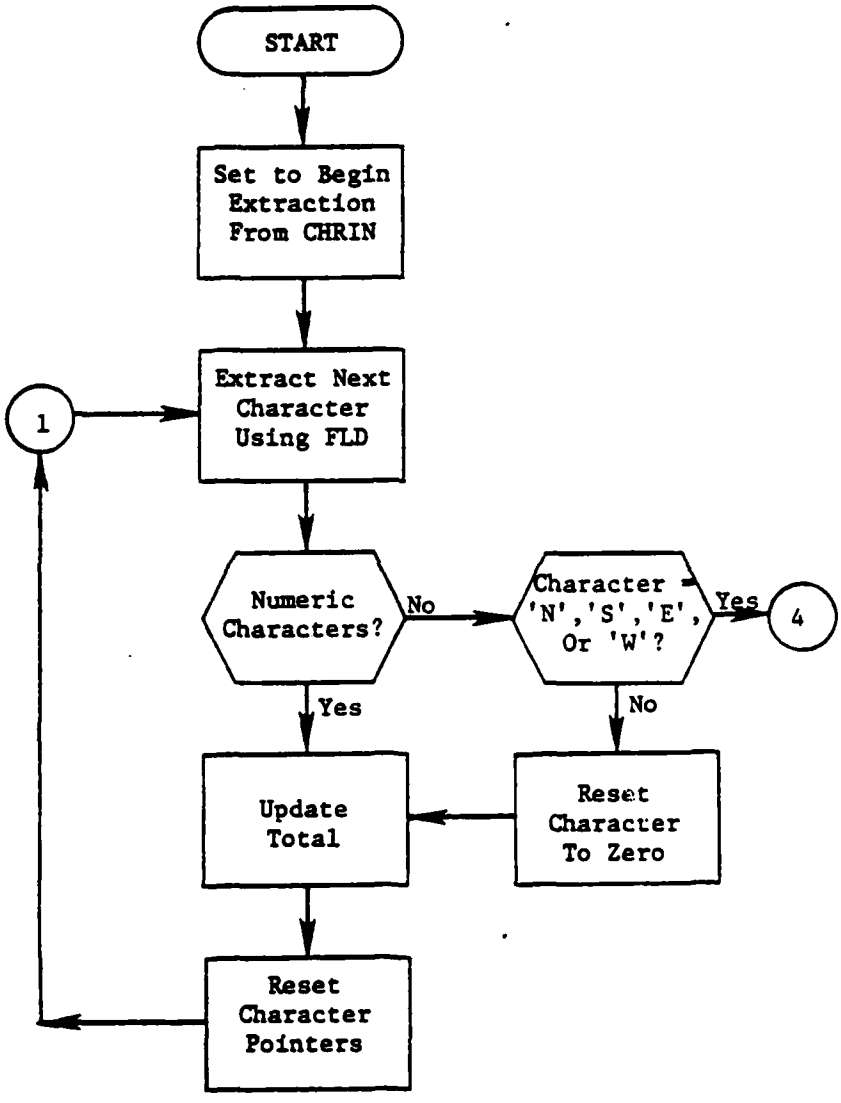


Figure 190. Function XLL (Part 1 of 2)

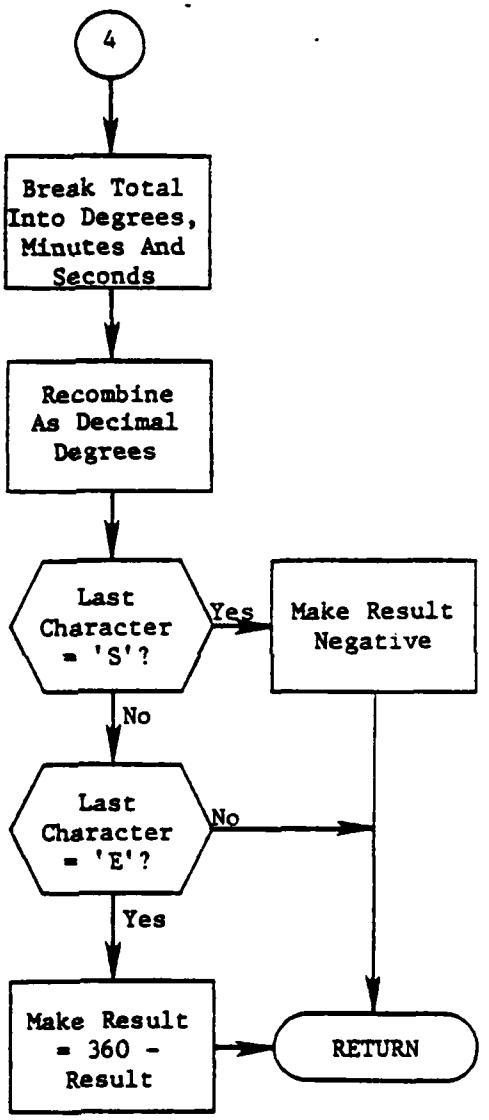


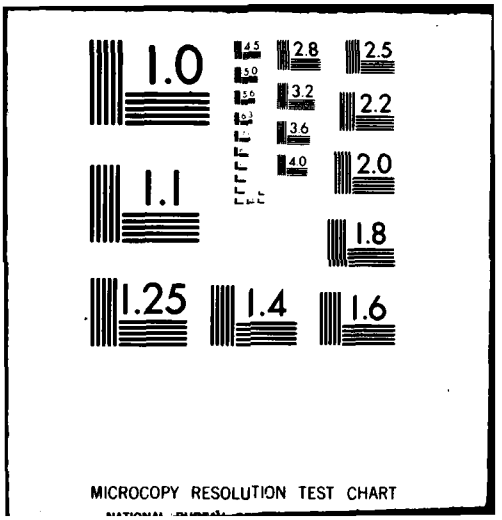
Figure 190. (Part 2 of 2)

THIS PAGE INTENTIONALLY LEFT BLANK

AD-A085 813 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON DC F/G 15/7
THE CCTC QUICK - REACTING GENERAL WAR GAMING SYSTEM (QUICK) PRO--ETC(U)
MAY 80
UNCLASSIFIED CCTC-CSM-MM-9-77-V1-CH6-3 NL

3 - 3
AD A085 813

END
DATE FILMED
8-80
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART

9.61 Subroutine XMATH

PURPOSE: To execute mathematical calculations

ENTRY POINTS: XMATH

FORMAL PARAMETERS:

X:	Array of internal variables to be used in calculations
BEGIN:	Index of first instruction code
END:	Index of end of calculation

COMMON BLOCKS: C30

SUBROUTINES CALLED: INSGET, IORFL, OFVAL, UNCODE

Method:

This subroutine is best understood by reference to figure 191. Basically the instructions retrieved from INSGET starting with BEGIN and ending with END are executed. In the process, values will be stored in X which is used for internal variables. The current value is maintained locally in Q and the new value is stored locally in R. For each instruction a branch (IBR) is set and then the remainder of the instruction is used to determine the value of R. Then the branch is made and whatever operation on Q and R is called for is carried out with the result placed in Q.

APPENDIX A

COP EXTERNAL COMMON BLOCKS

This appendix contains those common blocks used to communicate between the COP and related modules of the QUICK system. The appendix contains the following common blocks:

- o C10 IDS Communications control block
- o C15 Header reference codes
- o C20 Record type name and number
- o C30 Data base attributes
- o C40 Utility table
- o C50 Display table
- o ERRCOM IDS error code control block
- o INS Input instruction code buffer
- o IPGT Input card image buffer
- o OOPS Error flag
- o QC Data Module Quality Control
- o STRING Interpreted input character string
- o VBINDX Data Module Quality Control Cross-reference

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
C10		IDS communications control block
	IREFZ	Binary reference code, updated whenever IDS action takes place
	MQ(2)	(Not used)
	IRECTP	Record type number, updated whenever IDS action takes place
	NQ	(Not used)
	ERCODE	IDS error code
	NQ2	(Not used)
C15	HEADRF	Header reference code. BCD representation. Variable is type character *8
C20		Contains values for record type INDRCT
	RNAME	Record type name
	RNUMB	Record type number
C30	MAIN(306)	Contains all data base attributes. For precise attributes definition and their addresses within the array see Users Manual, Volume I
C40		Utility table (TABLEZ)
	TABREF	TABLEZ BCD reference code (type is character*8)
	TABLE(100)	Body of table
C50		Display table (DISPDT)
	DSPTAB(100)	Body of table (see section 6)
ERRCOM	NORMAC	Action to take if error code is not in list CHEKS (ABORT, FLAG, PASS)
	CHCKAC	Action to take if error code is in list CHEKS
	ERUNIT	Unit on which to print error message
	NUMCHK	Number of error codes in CHEKS list
	CHEKS(30)	List of error codes to check

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
INS	INSBUF(100)	Input instruction code buffer
	INSREF(50)	Reference codes for input instruction code tables. (Type is character*8)
	INSTBS	Number of instruction code tables
	INSTCR	Index number of instruction code table currently in buffer
IPQT	POINTER	Points to next character of input card image
	INBUF(80)	Input card image. Each letter is stored in separate word
	SPCIAL	Switch which controls use of '+' and '-' (true if preceding input string was an operator; else false)
	ENDSW	End of input switch
OOPS	ERROR	Error flag, causes COP to check only syntax when on
QC		Used to pass statistics on data module quality control
	ITCNT	Count of the number of create, change and delete transactions processed by the data module
	IRACNT	The number of records affected by a single transaction
	IRTA(30)	An array containing unique record types affected by a single transaction
	LCNT	The number of lines to be printed per page on the output listing
	IRTYP	A variable used to keep track of unique record types
	SYDE	Represents the side which is affected by the transaction (Blue or Red)

BLOCK	VARIABLE OR ARRAY	DESCRIPTION
STRING	TYPE	Current input string's type (not the attribute of the same spelling). If output from subroutine GETSTR results are: =1, operator =2, long string delineates =9, alphabetic =10, numeric
	VALUE	Identifying value, depends on type of string
	NUMBER	Floating point value if string type =10
	ALPHA	Character string being interpreted. Will be blank if string type =1

APPENDIX B

EXECUTABLE JOB CONTROL LANGUAGE (JCL)
QUICK SYSTEM

The QUICK system executes from a temporary H* which is assembled from object decks (C*s) which are stored on permfiles. Figure 194 contains the necessary JCL to execute the QUICK system from these object decks. The object files COP, BOOT, ERRFND, and INPTRN are required for every QUICK run. Other modules should be selected as needed.

A composite listing of the JCL required to compile the QUICK source decks to create the object decks is listed in figure 195. each section between $ LINK cards can be compiled separately.

The QUICK utility programs are contained in the QUICK utility library file. JCL to create this file is listed in figure 196.

```
$       IDENT   18205ID/3J/J584,QUICK RUN  JCL ,C314
$       USERID  QJ3NI31405$PASSWORD
$       PARAM   12345
$       SELECT  6311DP00/PERFORM/RESTORE
$       PRMFL   QD,R/W,R,6311DPOJ/QUIK/COP/IDS
$       OPTION  FORTRAN
$       LOWLOAD
$       LIBRARY UL,PL
$       ENTRY   ......
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/COP
$       IDS     DECK
$       FILE    *3,X1R,100R
$       PRMFL   I*,W,S,6341DPJO/QUIK/OBJECT/QDATA
$       USE     .QMAX/1/,.QAREA/3126/,.QMIN/1/,.FRRD.
$       LINK    BOOTT
$       PRMFL   C*,W,S,6341DPJO/QUIK/OBJECT/BOOT
$       LINK    TABSTR,BOJTT
$       USE     TABL2/BOE/
$       LINK    ERRF
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/ERRFND
$       LINK    INPT,ERRF
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/INPTRN
$       LINK    MODULE,TABSTR
$       LINK    JLM,MODULE
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/JLM
$       LINK    ASSI
$       PRMFL   C*,W,S,6341DPJO/QUIK/OBJECT/ASSIGN
$       LINK    SELE,ASSI
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/SELECT
$       LINK    ASTE,SELE
$       PRMFL   C*,W,S,6341DPJO/QUIK/OBJECT/ASTERISK
$       LINK    MODULE,JLM
$       LINK    DATA,MODULE
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/DATA
$       LINK    DATADL
$       PRMFL   C*,W,S,6341DPJO/QUIK/OBJECT/DELETE
$       LINK    DATACH,DATADL
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/CHANGE
$       LINK    DATACR,DATACH
$       PRMFL   C*,W,S,6341DPJO/QUIK/OBJECT/CREATE
$       LINK    MODULE,DATA
$       LINK    DBMOD,MODULE
$       PRMFL   C*,W,S,6341DPJO/QUIK/OBJECT/DBMOD
$       LINK    MODULE,DBMOD
$       LINK    INDXER,MODULE
$       PRMFL   C*,W,S,6341DPJO/QUIK/OBJECT/INDEXER
$       LINK    MODULE,INDXER
$       LINK    PLANS,MODULE
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/PLANSET
$       LINK    MODULE,PLANS
$       LINK    PREP,MODULE
$       PRMFL   C*,W,S,6341DPJO/QUIK/OBJECT/PREPALOC
$       LINK    MODULE,PREP
$       LINK    EDIT,MODULE
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/EDITDB
```

Figure 194. QUICK Execution JCL From Object Decks (Part 1 of 4)

```
$       LINK    ENORMA
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/NORMAL
$       LINK    EGENED,ENORMA
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/GENEDIT
$       LINK    EPROCE,EGENED
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/PROCEDIT
$       LINK    ECOUNT,EPROCE
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/COUNTS
$       LINK    MODULE,EDIT
$       LINK    REPORT,MODULE
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/REPORT
$       LINK    RPTDSN
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/DESIGN
$       LINK    RPTALT,RPTDSN
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/ALTER
$       LINK    RPTDMK,RPTALT
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/DSPMAK
$       LINK    RPTPRN,RPTDMK
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/PRINT
$       LINK    MODULE,REPORT
$       LINK    SRM,MODULE
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/SRM
$       LINK    MODULE,SRM
$       LINK    EIM,MODULE
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/EIM
$       LINK    BSIDAC
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/SIDAC
$       LINK    BOTHER,BSIDAC
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/BLDOTH
$       LINK    BTABLE,BOTHER
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/TABLE
$       LINK    PLOTTT,BTABLE
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/PLOTDATA
$       LINK    PLOTIT,PLOTTT
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/PLOTIT
$       LINK    MODULE,EIM
$       LINK    ALOC,MODULE
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/ALOC
$       LINK    ALCINT
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/ALCINT
$       LINK    ALCMUL,ALCINT
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/ALCMUL
$       LINK    FGD
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/FGD
$       LINK    SGD,FGD
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/SGD
$       LINK    STAL,SGD
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/STAL
$       LINK    DEFAL,STAL
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/DEFAL
$       LINK    MODULE,ALOC
$       LINK    EVAL,MODULE
$       PRMFL   C*,W,S,6341DPJ0/QUIK/OBJECT/EVALALOC
$       LINK    MODULE,EVAL
$       LINK    DGZSEL,MODULE
```

Figure 194. (Part 2 of 4)

```
$       PRMFL   C*,W,S,6341DP30/QUIK/OBJECT/ALOCOUT
$       LINK    OFFSET
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/OFFSET
$       LINK    ASGSET,OFFSET
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/ASGSET
$       LINK    MINIO,ASGSET
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/MINIO
$       LINK    MODULE,DGZSEL
$       LINK    POST,MODULE
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/POSTALOC
$       PRMFL   C*,W,S,6341DP30/QUIK/OBJECT/POSTAL02
$       LINK    MODULE,POST
$       LINK    FOOT,MODULE
$       PRMFL   C*,W,S,6341DP30/QUIK/OBJECT/FOOTPRNT
$       LINK    OPTS
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/FOOTOPTS
$       LINK    SETS,OPTS
$       PRMFL   C*,W,S,6341DP30/QUIK/OBJECT/FOOTSETS
$       LINK    PLAN,SETS
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/FOOTPLAN
$       LINK    ASGN,PLAN
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/FOOTASGN
$       LINK    MODULE,FOOT
$       LINK    MDUMP,MODULE
$       PRMFL   C*,W,S,6341DP30/QUIK/OBJECT/MIRVDUMP
$       LINK    MODULE,MDUMP
$       LINK    PLANO,MODULE
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/PLANO
$       LINK    PLNT
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/PLNT
$       LINK    INTR,PLNT
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/INTR
$       LINK    TANK,INTR
$       PRMFL   C*,W,S,6341DP30/QUIK/OBJECT/TANK
$       LINK    MODULE,PLANO
$       LINK    DMAKE,MODULE
$       PRMFL   C*,W,S,6341DP00/QUIK/OBJECT/DATAMAKE
$       LINK    MODULE,DMAKE
$       EXECUTE DUMP,DEBUG,VREST,JREST
$       FILE    M*,H1S,75R
$       FFILE   P*,LGU/(06,42,43,11,12,13)
$       PRMFL   UL,R,R,6341DP00/QUIK/LIBRARY/UTIL
$       PRMFL   PL,R,S,LIBRARY/PLOTLIB
$       PRMFL   QD,R/W,R,6341DP00/QUIK/COP/IDS
$       FILE    19,X19S,50L
$       FFILE   19,NBUFFS/2,NOSLEW
$       DATA    I*
$       TAPE9   15,T15D,,61243,,OUTPUT-TAPE
$       TAPE9   32 T32D,,18943,,RESTORE-TAPE
$       TAPE9   33,T33D,,19740,,SAVE-TAPE
$       TAPE9   35,T35D,,62433,,OUTPUT-TAPE
$       TAPE9   36,T36D,,55741,,OUTPUT-TAPE
$       LIMITS  75,78K,-4K,51K
$       FILE    25,X25S,150R
$       FILE    21,X21S,50L
```

Figure 194. (Part 3 of 4)

```
$       FFILE   21,NBUFFS/2,NOSLEW
$       FILE    22,X22S,SOL
$       FFILE   22,NBUFFS/2,NOSLEW
$       FILE    23,X23S,SOL
$       FFILE   23,NBUFFS/2,NOSLEW
$       FILE    24,X24S,SOL
$       FFILE   24,NBUFFS/2,NOSLEW
$       PARAM   54321
$       SELECT  634IDP00/PERFORM/SAVE
$       PRMFL   GO,R/W,R,6311DP0J/QUIK/COP/IDS
$       ENDJOB
```

Figure 194. (Part 4 of 4)

```
$       IDENT   1820510/30/0577,QUICK COMPILE JCL
$       USERID  DJ3NI314C5$PASSWORD
$       OPTION  FORTRAN
$       LOWLOAD
$       LIBRARY UL,PL
$       ENTRY   ......
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/COP
$       SELECT  631IDPX0/DUAL/SOURCE/COP/COP
$       SELECT  631IDPX0/DUAL/SOURCE/COP/INSPUT
$       SELECT  631IDPX0/DUAL/SOURCE/COP/MODGET
$       SELECT  631IDPX0/DUAL/SOURCE/COP/HDFND
$       SELECT  631IDPX0/DUAL/SOURCE/COP/INICOP
$       SELECT  631IDPX0/DUAL/SOURCE/COP/ERPROC
$       SELECT  631IDPX0/DUAL/SOURCE/COP/BANNER
$       SELECT  631IDPX0/DUAL/SOURCE/COP/INPRIN
$       IDS     DECK
$       LIMITS  10,38K,,23K
$       FILE    *3,X1R,100R
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/QDATA
$       SELECT  631IDPX0/DUAL/SOURCE/COP/QDATA
$       SELECT  631IDPX0/DUAL/SOURCE/COP/QDATB
$       USE     .QMAX/1/,.QAREA/3126/,.QMIN/1/,.FRRD.
$       LINK    BOOTT
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/BOOT
$       SELECT  631IDPX0/DUAL/SOURCE/COP/BOOT
$       LINK    TABSTR,BOOTT
$       USE     TABLZ/808/
$       LINK    ERRF
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/ERRFND
$       SELECT  631IDPX0/DUAL/SOURCE/COP/ERRFND
$       SELECT  631IDPX0/DUAL/SOURCE/COP/WEBSTR
$       SELECT  631IDPX0/DUAL/SOURCE/COP/SYNTAX
$       SELECT  631IDPX0/DUAL/SOURCE/COP/TABINS
$       SELECT  631IDPX0/DUAL/SOURCE/COP/LNGSTR
$       LINK    INPT,ERRF
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/INPTRN
$       SELECT  631IDPX0/DUAL/SOURCE/COP/INPTRN
$       SELECT  631IDPX0/DUAL/SOURCE/COP/INMATH
$       SELECT  631IDPX0/DUAL/SOURCE/COP/DELTAB
$       SELECT  631IDPX0/DUAL/SOURCE/COP/PARLEV
$       SELECT  631IDPX0/DUAL/SOURCE/COP/TABGET
$       SELECT  631IDPX0/DUAL/SOURCE/COP/LINEIO
$       LINK    MODULE,TABSTR
$       LINK    JLM,MODULE
$       FORTY   MAP,XREF,DECK
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/JLM
$       LIMITS  01,26K,,5K
```

Figure 195. QUICK Compilation JCL (Part 1 of 10)

```
$       SELECT  631IDPX0/DUAL/SOURCE/JLM/JLM
$       LINK    ASSI
$       FORTY   MAP,XREF,DECK
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/ASSIGN
$       LIMITS  01,30K,,5K
$       SELECT  631IDPX0/DUAL/SOURCE/JLM/ASSIGN
$       SELECT  631IDPX0/DUAL/SOURCE/JLM/ALPHAS
$       SELECT  631IDPX0/DUAL/SOURCE/JLM/PLAYERS
$       SELECT  631IDPX0/DUAL/SOURCE/JLM/TOPRINT
$       LINK    SELE,ASSI
$       FORTY   MAP,XREF,DECK
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/SELECT
$       LIMITS  01,28K,,9K
$       SELECT  631IDPX0/DUAL/SOURCE/JLM/SELECT
$       SELECT  631IDPX0/DUAL/SOURCE/JLM/DEFAULT
$       SELECT  631IDPX0/DUAL/SOURCE/JLM/ADTOBASE
$       SELECT  631IDPX0/DUAL/SOURCE/JLM/SAMSET
$       SELECT  631IDPX0/DUAL/SOURCE/JLM/KRUNCH
$       LINK    ASTE,SELE
$       FORTY   MAP,XREF,DECK
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/ASTERISK
$       LIMITS  05,24K,,5K
$       SELECT  631IDPX0/DUAL/SOURCE/JLM/ASTERISK
$       LINK    MODULE,JLM
$       LINK    DATA,MODULE
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/DATA
$       SELECT  631IDPX0/DUAL/SOURCE/DATA/ENTMOD
$       LINK    DATADL
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/DELETE
$       SELECT  631IDPX0/DUAL/SOURCE/DATA/DELETE
$       LINK    DATACH,DATADL
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/CHANGE
$       SELECT  631IDPX0/DUAL/SOURCE/DATA/CHANGE
$       SELECT  631IDPX0/DUAL/SOURCE/DATA/VALPUT
$       SELECT  631IDPX0/DUAL/SOURCE/DATA/DESSCH
$       SELECT  631IDPX0/DUAL/SOURCE/DATA/NXTDES
$       LINK    DATACR,DATACH
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/CREATE
$       SELECT  631IDPX0/DUAL/SOURCE/DATA/CREATE
$       SELECT  631IDPX0/DUAL/SOURCE/DATA/VALPUT
$       LINK    MODULE,DATA
$       LINK    DBMOD,MODULE
$       FORTY   MAP,XREF,DECK
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/DBMOD
$       LIMITS  01,31K,,5K
$       SELECT  631IDPX0/DUAL/SOURCE/DBMOD/DBMOD
$       SELECT  631IDPX0/DUAL/SOURCE/DBMOD/DESTAB
```

Figure 195. (Part 2 of 10)

```
$       LINK    MODULE,DBMOD
$       LINK    INDXER,MODULE
$       FORTY   MAP,XREF,DECK
$       LIMITS  02,35K,,25K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/INDEXER
$       SELECT  6311DPX0/DUAL/SOURCE/INDEXER/INDEXER
$       SELECT  6311DPX0/DUAL/SOURCE/INDEXER/COMPLEX
$       SELECT  6311DPX0/DUAL/SOURCE/INDEXER/SETVAL
$       SELECT  6311DPX0/DUAL/SOURCE/INDEXER/CRTBLE
$       LINK    MODULE,INDXER
$       LINK    PLANS,MODULE
$       FORTY   MAP,XREF,DECK
$       LIMITS  02,44K,,30K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/PLANSET
$       SELECT  6311DPX0/DUAL/SOURCE/PLANSET/PLANSET
$       SELECT  6311DPX0/DUAL/SOURCE/PLANSET/GRPEM
$       SELECT  6311DPX0/DUAL/SOURCE/PLANSET/SRTTGT
$       SELECT  6311DPX0/DUAL/SOURCE/PLANSET/CALCOMP
$       SELECT  6311DPX0/DUAL/SOURCE/PLANSET/ADJUSTGP
$       SELECT  6311DPX0/DUAL/SOURCE/PLANSET/PRINTGP
$       SELECT  6311DPX0/DUAL/SOURCE/PLANSET/TANKER
$       LINK    MODULE,PLANS
$       LINK    PREP,MODULE
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,35K,,14K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/PREPALOC
$       SELECT  6311DPX0/DUAL/SOURCE/PREPALOC/PREPALOC
$       SELECT  6311DPX0/DUAL/SOURCE/PREPALOC/FIXWEP
$       SELECT  6311DPX0/DUAL/SOURCE/PREPALOC/CHGBAS
$       SELECT  6311DPX0/DUAL/SOURCE/PREPALOC/GEOPREP
$       SELECT  6311DPX0/DUAL/SOURCE/PREPALOC/GEOIN
$       SELECT  6311DPX0/DUAL/SOURCE/PREPALOC/WEPIN
$       SELECT  6311DPX0/DUAL/SOURCE/PREPALOC/SETRD
$       SELECT  6311DPX0/DUAL/SOURCE/PREPALOC/WEPPREP
$       SELECT  6311DPX0/DUAL/SOURCE/PREPALOC/PRNPRP
$       LINK    MODULE,PREP
$       LINK    EDIT,MODULE
$       FORTY   MAP,XREF,DECK
$       LIMITS  02,30K,,1JK
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/EDITDB
$       SELECT  6311DPX0/DUAL/SOURCE/EDITDB/EDITDB
$       LINK    ENORMA
$       FORTY   MAP,XREF,DECK
$       LIMITS  01,30K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/NORMAL
$       SELECT  6311DPX0/DUAL/SOURCE/EDITDB/NORMAL
$       LINK    EGENED,ENORMA
$       FORTY   MAP,XREF,DECK
$       LIMITS  01,30K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/GENEDIT
$       SELECT  6311DPX0/DUAL/SOURCE/EDITDB/GENEDIT
$       SELECT  6311DPX0/DUAL/SOURCE/EDITDB/BUILDTAB
$       SELECT  6311DPX0/DUAL/SOURCE/EDITDB/SETFLD
$       SELECT  6311DPX0/DUAL/SOURCE/EDITDB/SWITH
$       SELECT  6311DPX0/DUAL/SOURCE/EDITDB/FORMLOC
```

Figure 195. (Part 3 of 10)

```
$       LINK    EPROCE,EGENED
$       FORTY   MAP,XREF,DECK
$       LIMITS  01,30K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/PROCEDIT
$       SELECT  6311DPX0/DUAL/SOURCE/EDITDB/PROCEDIT
$       SELECT  6311DPX0/DUAL/SOURCE/EDITDB/XWITH
$       LINK    ECOUNT,EPROCE
$       FORTY   MAP,XREF,DECK
$       LIMITS  01,30K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/COUNTS
$       SELECT  6311DPX0/DUAL/SOURCE/EDITDB/COUNTS
$       LINK    MODULE,EDIT
$       LINK    REPORT,MODULE
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/REPORT
$       SELECT  6311DPX0/DUAL/SOURCE/REPORT/ENTMOD
$       SELECT  6311DPX0/DUAL/SOURCE/REPORT/DSPPUT
$       SELECT  6311DPX0/DUAL/SOURCE/REPORT/TABMNT
$       LINK    RPTDSN
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/DESIGN
$       SELECT  6311DPX0/DUAL/SOURCE/REPORT/DESIGN
$       LINK    RPTALT,RPTDSN
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/ALTER
$       SELECT  6311DPX0/DUAL/SOURCE/REPORT/ALTER
$       LINK    RPTDMK,RPTALT
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/DSPMAK
$       SELECT  6311DPX0/DUAL/SOURCE/REPORT/DSPMAK
$       LINK    RPTPRN,RPTDMK
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/PRINT
$       SELECT  6311DPX0/DUAL/SOURCE/REPORT/PRINT
$       SELECT  6311DPX0/DUAL/SOURCE/REPORT/XDEFN
$       SELECT  6311DPX0/DUAL/SOURCE/REPORT/PRNATD
$       LINK    MODULE,REPORT
$       LINK    SRM,MODULE
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/SRM
$       SELECT  6311DPX0/DUAL/SOURCE/SRM/SRM
$       LINK    MODULE,SRM
$       LINK    EIM,MODULE
$       LIMITS  10,32K,,20K
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/EIM
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/ENTMOD
$       LINK    BSIDAC
```

Figure 195. (Part 4 of 10)

```
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/SIDAC
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/SIDAC
$       LINK    BOTHER,BSIDAC
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/BLDOTH
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/BLDOTH
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/XEDEFN
$       LINK    BTABLE,BOTHER
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/TABLE
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/TABLE
$       LINK    PLOTTT,BTABLE
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/PLOTDATA
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/PLOTDATA
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/PICS
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/PROJCT
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/HOUSKEEP
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/PIECEIT
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/PIECENUM
$       LINK    PLOTIT,PLOTTT
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/PLOTIT
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/PLOTIT
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/INTRPL
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/PLBLOFF
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/SUBPLOT
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/SUBREAD
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/FNDSRT
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/PLOTINIT
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/PICS
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/PROJCT
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/HOUSKEEP
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/PIECEIT
$       SELECT  6311DPX0/DUAL/SOURCE/EIM/PIECENUM
$       LINK    MODULE,EIM
$       LINK    ALOC,MODULE
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/ALOC
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/ALOC
$       LINK    ALCINT
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,20K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/ALCINT
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/INITAL
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/CNCLST
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/DATGRP
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/FLOCRS
```

Figure 195. (Part 5 of 10)

```
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/MRVRST
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/PRNPUT
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/RDMUL
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/RDPRNZ
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/RDSET
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/RDSMAT
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/RNGALT
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/SETABLE
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/TIMEPRT
$       LINK    ALCMUL,ALCINT
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,20K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/ALCMUL
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/MULCON
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/ADDSAL
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/ASGOUT
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/BOMPRM
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/PRNTALL
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/PRNTCON
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/PRNTNOW
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/TABLEMUP
$       LINK    FGD
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,20K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/FGD
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/FRSTGD
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/CRDCAL
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/FLGCHK
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/INICRD
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/RTAPCK
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/PKCALC
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/PRNTOF
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/RECON
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/SETPAY
$       LINK    SGD,FGD
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/SGD
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/SCNDGD
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/RTAPCK
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/RECON
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/SETPAY
$       LINK    STAL,SGD
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,20K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/STAL
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/STALL
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/FORMATS
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/FMUP
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/LAMGET
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/PREMIUMS
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/PRNTOS
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/SALVAL
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/WAD
$       SELECT  6311DPX0/DUAL/SOURCE/ALOC/WADOUT
```

Figure 195. (Part 6 of 10)

```
$       LINK    DEFAL,STAL
$       FORTY   MAP,XREF,DECK
$       LIMITS  10,32K,,10K
$       PRMFL   C*,W,S,631IDPXO/DUAL/OBJECT/DEFAL
$       SELECT  631IDPXO/DUAL/SOURCE/ALOC/DEFALOC
$       SELECT  631IDPXO/DUAL/SOURCE/ALOC/FMUP
$       SELECT  631IDPXO/DUAL/SOURCE/ALOC/LAMGET
$       SELECT  631IDPXO/DUAL/SOURCE/ALOC/PREMIUMS
$       SELECT  631IDPXO/DUAL/SOURCE/ALOC/PRNTOD
$       SELECT  631IDPXO/DUAL/SOURCE/ALOC/RESVAL
$       SELECT  631IDPXO/DUAL/SOURCE/ALOC/SALVAL
$       LINK    MODULE,ALOC
$       LINK    EVAL,MODULE
$       FORTY   MAP,XREF,DECK
$       PRMFL   C*,W,S,631IDPXO/DUAL/OBJECT/EVALALOC
$       LIMITS  01,40K,,10K
$       SELECT  631IDPXO/DUAL/SOURCE/EVALALOC/EVALALOC
$       SELECT  631IDPXO/DUAL/SOURCE/EVALALOC/EVALPLAN
$       SELECT  631IDPXO/DUAL/SOURCE/EVALALOC/EVAL2
$       SELECT  631IDPXO/DUAL/SOURCE/EVALALOC/PREVAL
$       SELECT  631IDPXO/DUAL/SOURCE/EVALALOC/SSSPCALC
$       SELECT  631IDPXO/DUAL/SOURCE/EVALALOC/TGTMODIF
$       SELECT  631IDPXO/DUAL/SOURCE/EVALALOC/WPNMODIF
$       LINK    MODULE,EVAL
$       LINK    DGZSEL,MODULE
$       FORTY   MAP,XREF,DECK
$       LIMITS  02,35K,,30K
$       PRMFL   C*,W,S,631IDPXO/DUAL/OBJECT/ALOCOUT
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/ALOCOUT
$       LINK    OFFSET
$       FORTY   MAP,XREF,DECK
$       LIMITS  02,35K,,30K
$       PRMFL   C*,W,S,631IDPXO/DUAL/OBJECT/OFFSET
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/COMPRESS
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/CUMINV
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/DGZ
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/ERGOT1
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/FINDMIN
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/F2BMIN
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/GRADF
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/MOVE
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/PERTBLD
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/PROCCOMP
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/SEECALC
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/SEEINPUT
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/VAL
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/VMARG
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/WEPGET
$       LINK    ASGSET,OFFSET
$       FORTY   MAP,XREF,DECK
$       LIMITS  02,35K,,30K
$       PRMFL   C*,W,S,631IDPXO/DUAL/OBJECT/ASGSET
$       SELECT  631IDPXO/DUAL/SOURCE/ALOCOUT/SUMPRN
$       LINK    MINIO,ASGSET
$       FORTY   MAP,XREF,DECK
```

Figure 195. (Part 7 of 10)

```
$       LIMITS  02,35K,,33K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/MINIO
$       SELECT  6311DPX0/DUAL/SOURCE/ALOCOUT/MINIOUT
$       SELECT  6311DPX0/DUAL/SOURCE/ALOCOUT/FINDTIME
$       SELECT  6311DPX0/DUAL/SOURCE/ALOCOUT/INFORM
$       LINK    MODULE,DGZSEL
$       LINK    POST,MODULE
$       FORTY   DECK,MAP,XREF
$       LIMITS  03,40K,,25K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/POSTALOC
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/POSTALOC
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/GENRAID
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/GETGROUP
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/GETSORT
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/PRERAID
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/OUTSRT
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/PRINTIT
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/PRNTF
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/SETFLAG
$       FORTY   DECK,MAP,XREF
$       LIMITS  03,40K,,25K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/POSTAL02
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/CENTROID
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/CHGPLAN
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/CORRPARM
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/DIFF
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/DUMPSRT
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/EVALB
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/EVALOA
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/EVALOB
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/FLTPLAN
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/FLTROUTE
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/INITOPT
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/INPOTGT
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/NEXTFLT
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/NOCORR
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/OPTRAID
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/OUTPOTGT
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/SORTOPT
$       SELECT  6311DPX0/DUAL/SOURCE/POSTALOC/TGTASGN
$       LINK    MODULE,POST
$       LINK    FOOT,MODULE
$       FORTY   DECK,MAP,XREF
$       LIMITS  01,40K,-4K,15K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/FOOTPRNT
$       SELECT  6311DPX0/DUAL/SOURCE/FOOTPRNT/FOOTPRNT
$       LINK    OPTS
$       FORTY   DECK,MAP,XREF
$       LIMITS  01,40K,-4K,15K
$       PRMFL   C*,W,S,6311DPX0/DUAL/OBJECT/FOOTOPTS
$       SELECT  6311DPX0/DUAL/SOURCE/FOOTPRNT/TABLINPT
$       SELECT  6311DPX0/DUAL/SOURCE/FOOTPRNT/MKAOS
$       SELECT  6311DPX0/DUAL/SOURCE/FOOTPRNT/PRAOS
$       SELECT  6311DPX0/DUAL/SOURCE/FOOTPRNT/PRINSETS
$       SELECT  6311DPX0/DUAL/SOURCE/FOOTPRNT/TAOS
```

Figure 195. (Part 8 of 10)

```
$       LINK    SETS,OPTS
$       FORTY   DECK,MAP,XREF
$       LIMITS  01,40K,-4K,15K
$       PRMFL   C*,W,S,631IDPXO/DUAL/OBJECT/FOOTSETS
$       SELECT  631IDPXO/DUAL/SOURCE/FOOTPRNT/NEWCOOR
$       SELECT  631IDPXO/DUAL/SOURCE/FOOTPRNT/SETDATA
$       LINK    PLAN,SETS
$       FORTY   DECK,MAP,XREF
$       LIMITS  01,40K,-4K,15K
$       PRMFL   C*,W,S,631IDPXO/DUAL/OBJECT/FOOTPLAN
$       SELECT  631IDPXO/DUAL/SOURCE/FOOTPRNT/AXES
$       SELECT  631IDPXO/DUAL/SOURCE/FOOTPRNT/DRIVER
$       SELECT  631IDPXO/DUAL/SOURCE/FOOTPRNT/ELLIPSE
$       SELECT  631IDPXO/DUAL/SOURCE/FOOTPRNT/PATHFIND
$       SELECT  631IDPXO/DUAL/SOURCE/FOOTPRNT/XAOS
$       LINK    ASGN,PLAN
$       FORTY   DECK,MAP,XREF
$       LIMITS  01,40K,-4K,15K
$       PRMFL   C*,W,S,631IDPXO/DUAL/OBJECT/FOOTASGN
$       SELECT  631IDPXO/DUAL/SOURCE/FOOTPRNT/MISASGN
$       LINK    MODULE,FOOT
$       LINK    MDUMP,MODULE
$       FORTY   MAP,XREF,DECK
$       LIMITS  01,30K,-4K,15K
$       PRMFL   C*,W,S,631IDPXO/DUAL/OBJECT/MIRVDUMP
$       SELECT  631IDPXO/DUAL/SOURCE/MIRVDUMP/MIRVDUMP
$       SELECT  631IDPXO/DUAL/SOURCE/MIRVDUMP/GETGT
$       SELECT  631IDPXO/DUAL/SOURCE/MIRVDUMP/NEWCORD
$       SELECT  631IDPXO/DUAL/SOURCE/MIRVDUMP/DRIVER
$       SELECT  631IDPXO/DUAL/SOURCE/MIRVDUMP/COMBO
$       SELECT  631IDPXO/DUAL/SOURCE/MIRVDUMP/FOOTCHCK
$       SELECT  631IDPXO/DUAL/SOURCE/FOOTPRNT/SETDATA
$       SELECT  631IDPXO/DUAL/SOURCE/FOOTPRNT/AXES
$       SELECT  631IDPXO/DUAL/SOURCE/FOOTPRNT/XAOS
$       LINK    MODULE,MDUMP
$       LINK    PLANO,MODULE
$       FORTY   MAP,XREF,DECK
$       PRMFL   C*,W,S,631IDPXO/DUAL/OBJECT/PLANO
$       SELECT  631IDPXO/DUAL/SOURCE/PLANOUT/PLANOUT
$       SELECT  631IDPXO/DUAL/SOURCE/PLANOUT/CLINDATA
$       SELECT  631IDPXO/DUAL/SOURCE/PLANOUT/GEOGET
$       SELECT  631IDPXO/DUAL/SOURCE/PLANOUT/SNAPCON
$       SELECT  631IDPXO/DUAL/SOURCE/PLANOUT/WEPDATA
$       LINK    PLNT
$       LIMITS  05,30K,,15K
$       FORTY   MAP,XREF,DECK
$       LIMITS  20,40K,,40K
$       PRMFL   C*,W,S,631IDPXO/DUAL/OBJECT/PLNT
$       SELECT  631IDPXO/DUAL/SOURCE/PLANOUT/PLNTPLAN
$       SELECT  631IDPXO/DUAL/SOURCE/PLANOUT/ALTPLAN
$       SELECT  631IDPXO/DUAL/SOURCE/PLANOUT/ADJUST
$       SELECT  631IDPXO/DUAL/SOURCE/PLANOUT/ALTERR
$       SELECT  631IDPXO/DUAL/SOURCE/PLANOUT/CHGTIM
$       SELECT  631IDPXO/DUAL/SOURCE/PLANOUT/DECOYADD
$       SELECT  631IDPXO/DUAL/SOURCE/PLANOUT/DISTIME
```

Figure 195. (Part 9 of 10)

```
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/FINDME
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/FLTSORT
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/FLYPOINT
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/INITANK
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/KERPLUNK
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/LAUNCH
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/LNCHDATA
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/PLAN
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/PLANBOMB
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/PLANTMIS
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/POST
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/POSTLAUN
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/SNAPIT
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/SNAPOUT
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/SORBOMB
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/SWTCHALT
$       LINK    INTR,PLNT
$       FORTY   MAP,XREF,DECK
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/INTR
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/INTRFACE
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/ABOUT
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/FINDTIME
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/IAZIM
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/IFSET
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/IFUNCT
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/INFORM
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/NOP
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/PRNTOFFS
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/RDCLAUSE
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/STOUT
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/XSET
$       LINK    TANK,INTR
$       FORTY   MAP,XREF,DECK
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/TANK
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/PLANTANK
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/PRNTAB
$       SELECT  631IDPX0/DUAL/SOURCE/PLANOUT/VAM
$       LINK    MODULE,PLANO
$       LINK    DMAKE,MODULE
$       FORTY   MAP,XREF,DECK
$       LIMITS  01,35K,,17K
$       PRMFL   C*,W,S,631IDPX0/DUAL/OBJECT/DATAMAKE
$       SELECT  631IDPX0/DUAL/SOURCE/PLANSET/DATAMAKE
$       SELECT  631IDPX0/DUAL/SOURCE/INDEXER/CRTBLE
$       SELECT  631IDPX0/DUAL/SOURCE/INDEXER/COMPLEX
$       SELECT  631IDPX0/DUAL/SOURCE/PLANSET/CALCOMP
$       SELECT  631IDPX0/DUAL/SOURCE/PLANSET/SRTTGT
$       LINK    MODULE,DMAKE
$       ENDJOB
```

Figure 195. (Part 10 of 10)

```
$       FILEDIT SOURCE,OBJECT,INITIALIZE
$       LIMITS  1C,32K,,30K
$       FILE    K*,NULL
$       FILE    R*,X2S,10JL
$       DATA    *C,COPY,ENDFC
$       INCLUDE SOURCE,OBJECT
$       OPTION  FORTRAN
$       FORTY   MAP,XREF                                                ABOR
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/ABORT
$       FORTY   MAP,XREF                                                ACOS
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/ACOS
$       FORTY   MAP,XREF                                                ASIN
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/ASIN
$       FORTY   MAP,XREF                                                ATFN
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/ATFNDR
$       FORTY   MAP,XREF                                                ATN2
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/ATN2PI
$       FORTY   MAP,XREF                                                AZMU
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/AZMUTH
$       FORTY   MAP,XREF                                                CONV
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/CONVLL
$       FORTY   MAP,XREF                                                CINS
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/CINSGET
$       FORTY   MAP,XREF                                                DIFF
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/DIFFLONG
$       FORTY   MAP,XREF                                                DIST
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/DISTF
$       FORTY   MAP,XREF                                                DOTL
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/DOTLINE
$       FORTY   MAP,XREF                                                FINC
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/FINDCLAS
$       FORTY   MAP,XREF                                                FIND
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/FINDSIDE
$       FORTY   MAP,XREF                                                FORM
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/FORMAK
$       GMAP                                                            GETC
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/GETCLOCK
$       FORTY   MAP,XREF                                                GETN
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/GETNXT
$       FORTY   MAP,XREF                                                GETS
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/GETSTR
$       FORTY   MAP,XREF                                                EGTT
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/GETTAR
$       FORTY   MAP,XREF                                                GLOG
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/GLOG
$       FORTY   MAP,XREF                                                IGET
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/IGET
$       FORTY   MAP,XREF                                                IGEH
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/IGETHOB
$       FORTY   MAP,XREF                                                IMAX
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/IMAX
$       FORTY   MAP,XREF                                                INTE
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/INTERP
$       FORTY   MAP,XREF                                                INTR
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/INTERPGC
```

Figure 196. Utility Library Creation (Part 1 of 3)

```
$       FORTY   MAP,XREF                                                INTP
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/INTPIECE
$       FORTY   MAP,XREF                                                IORF
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/IORFL
$       FORTY   MAP,XREF                                                IPUT
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/IPUT
$       FORTY   MAP,XREF                                                ISOF
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/ISOFF
$       FORTY   MAP,XREF                                                ITLE
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/ITLE
$       FORTY   MAP,XREF                                                KEYM
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/KEYMAKE
$       FORTY   MAP,XREF                                                LINK
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/LINKUP
$       FORTY   MAP,XREF                                                LREO
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/LREORDER
$       FORTY   MAP,XREF                                                LUNC
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/LUNCH
$       FORTY   MAP,XREF                                                MAPE
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/MAPEDGE
$       FORTY   NLSTIN                                                  MISD
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/MISDATA
$       FORTY   MAP,XREF                                                OFVA
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/OFVAL
$       FORTY   MAP,XREF                                                ORDE
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/ORDER
$       FORTY   MAP,XREF                                                PRIM
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/PRIMHD
$       GMAP                                                            PRTM
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/PROCTIM
$       FORTY   MAP,XREF                                                PSRE
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/PSREC
$       FORTY   MAP,XREF                                                RNER
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/RANGER
$       FORTY   MAP,XREF                                                REOR
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/REORDER
$       FORTY   MAP,XREF                                                SETD
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/SETDEF
$       GMAP                                                            SETO
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/SETORD
$       FORTY   MAP,XREF                                                SETS
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/SETSCH
$       FORTY   MAP,XREF                                                SLOG
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/SLOG
$       FORTY   MAP,XREF                                                SORD
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/SORDID
$       FORTY   MAP,XREF                                                SSKP
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/SSKPC
$       GMAP                                                            SVTP
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/SVTP
$       GMAP                                                            SWTC
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/SWTCHT
$       FORTY   MAP,XREF                                                TIMM
$       SELECTD 6311DPX0/DUAL/SOURCE/UTIL/TIMEME
$       FORTY   NLSTIN                                                  TLIM
```

Figure 196. (Part 2 of 3)

```
$       SELECTD 631IDPXO/DUAL/SOURCE/UTIL/TGTLIM
$       FORTY   MAP,XREF                                                TOFM
$       SELECTD 631IDPXO/DUAL/SOURCE/UTIL/TOFM
$       FORTY   MAP,XREF                                                UNCO
$       SELECTD 631IDPXO/DUAL/SOURCE/UTIL/UNCODE
$       FORTY   MAP,XREF                                                VALF
$       SELECTD 631IDPXO/DUAL/SOURCE/UTIL/VALFND
$       FORTY   MAP,XREF                                                VALT
$       SELECTD 631IDPXO/DUAL/SOURCE/UTIL/VALTAR
$       FORTY   MAP,XREF                                                VLRD
$       SELECTD 631IDPXO/DUAL/SOURCE/UTIL/VLRADP
$       FORTY   MAP,XREF                                                XLLL
$       SELECTD 631IDPXO/DUAL/SOURCE/UTIL/XLL
$       FORTY   MAP,XREF                                                XMAT
$       SELECTD 631IDPXO/DUAL/SOURCE/UTIL/XMATH
$       FORTY   MAP,XREF                                                XWHE
$       SELECTD 631IDPXO/DUAL/SOURCE/UTIL/XWHERE
$       FORTY   MAP,XREF                                                ZTAN
$       SELECTD 631IDPXO/DUAL/SOURCE/UTIL/ZTAN
$       ENDEDIT
$       ENDCOPY *C
$       PROGRAM RANLIB
$       FILE    R*,X2R,10JL
$       PRMFL   A6,R/W,R,631IDPXJ/DUAL/LIBRARY/UTIL
$       BREAK
```

Figure 196. (Part 3 of 3)

APPENDIX C

PERFORM PROGRAM

This appendix contains maintenance information for the PERFORM program. PERFORM is an online program designed to generate remote job entry jobs for the QUICK system.

C.1 Purpose

PERFORM is an online interactive program which creates a file of Job Control Language (JCL) according to user directions. This file of JCL may be set up to perform any combination of the following functions:

- Run QUICK
- Initialize the I-D-S Data Base
- Recompile and recreate the QUICK utility subroutine library
- Recompile a module of QUICK.

C.2 Input

PERFORM is an interactive system and, therefore, obtains part of its input from user responses from the terminal. PERFORM also has three files which it uses to build the job stream JCL and a set of object (CANOF) and source (NEWCANOF) files which are used to select required source and object programs.

The file names for these files are slightly different in the production and development systems. Source decks are only contained in the development system. File names for the production system (UMC 634IDP00) are used whenever the appropriate file exists on the production system. Otherwise, the file name from the development system (UMC 631IDPX0) is used. Both the production and development systems include:

- A set of files each of which contains the JCL required to define a module and its linkage from object files. These files are under the catalog UMC/QUIK/COP/CANOF.

- A set of files each of which contains the JCL required to define a module and its linkage from source files. These files are under the catalog 631IDPX0/DUAL/COP/NEWCANOF only (i.e., not on production system).

- A file (UMC/PERFORM/VRBLIM) which details the various required limits for the system and which contains one record for each legal verb.

  - Column 1-8 Verb Name
  - Column 9-11 Maximum CPU Time
  - Column 12-14 Maximum Core Requirements
  - Column 15-17 Maximum Lines of Output

- A file (UMC/PERFORM/IDENT2) which details the currently recognized users of the QUICK system. It also contains information concerning the data files accessed by these users. For each user, the file contains two records plus additional records equal to the number of data files used.

  Record 1

Column	1-12	USERID
Column	13-24	Name used in PERFORM interactive output
Column	25	Number of files used
Column	26	User salutation parameter: 1 - Salutation suppressed 2 - Salutation activated

  Record 2

Column	1-48	IDENT card for user

  Record 3 and following

Column	1-4	Source subcatalog name (i.e., TEST, DUAL, QUIK)
Column	5-8	Number of pages in data file
Column	9-32	Data file name

- A file (UMC/PERFORM/SPFILE) which details an additional file whose description must be added to those normally found in the JCL. These descriptions are related to the verbs which the user has specified. Each additional file has one record.

Column	1-8	Verb
Column	9	Blank
Column	10-11	File code
Column	12	R for input file W for output file
Column	13-48	Description of file used in PERFORM output

C.3 Concept of Operation

PERFORM follows a series of steps at user direction. For details of the question and answer sequences, see CSM UM 9-77, Volume I. Based on the selected value of MODE, PERFORM generates JCL for the job stream by adding a series of file names from the appropriate /CANOF and/or /NEWCANOF catalogs. The RUN mode adds RESTORE or SAVE records to file THEJOB as desired. It then adds the standard CANOF catalog files required by all QUICK runs and the additional CANOF files requested by the user. The CANOF files contain a list of object decks required to execute the desired modules. The user then supplies a list of the verbs being used. From this list, PERFORM uses the information stored on file VRBLIM to compute the LIMITS card parameters. The user may alter these. The user is then requested to add data files and/or lines of input. Then the SPFILE is consulted to see if the user desires any special files. Finally, the LIMITS and other final cards are added and the user is instructed how to submit the job.

For INITIALIZE mode, an activity initializing the I-D-S data file is added to THEJOB. The program then continues in the RUN mode.

The COMPILE mode adds NEWCANOF catalog files to file THEJOB for the modules selected for recompiling. Two modules (ALOC and PLANOUT) have been divided into submodules for compilation purposes. The submodule names are displayed to the user when the basic module is encountered. NEWCANOF records are added to file THEJOB for the selected submodules.

Upon completion of the COMPILE mode, the user is asked if the RUN mode is also desired. If so, additional modules required for the run are added to file THEJOB as described in the RUN mode discussion.

If the COMPILE mode contained a submodule NEWCANOF, CANOF records for all other submodules are added to THEJOB to provide a complete set of object decks for that module.

C.4 Major Subroutines

PERFORM has two subroutines. READIN scans user input and converts any lower case ASCII letters to upper case ASCII letters. IDENT builds the job stream IDENT card and selects the desired data base file.

C.5 Common Blocks

PERFORM has only one common block, LINE. This block contains the array LINE(80) which is used to communicate with subroutine READIN. Each word of LINE contains one character of input.

C.6 Program PERFORM

PURPOSE: To build JCL through online interaction

ENTRY POINTS: .....

FORMAL PARAMETERS: None

COMMON BLOCKS: LINE

SUBROUTINES CALLED: READIN, IDENT, WRIDEN, FILSEL

CALLED BY: Compiled by TSS subsystem YFORT
Entered through TSS subsystem RUNY

Method:

First, file THEJOB is attached and the IDENT card with proper disposition and urgency is added to it. Next, the mode is requested. If a reply other than COMPILE is entered, a transfer is made to statement 260 (figure 197). If the reply is COMPILE, the program asks if TEST or DUAL catalog is desired. The proper version of common block C30 is then transferred to the program storage file for C30.

The program then asks for the list of modules to be recompiled. A reply of HELP will list the modules. A key is set for each prestored module selected. If the module name is not prestored, the user is asked if recompilation of the unknown module is desired. If it is, the module name is stored and the corresponding key is set. Modules ALOC and PLANOUT have been divided into submodules for compilation purposes. When one of these is encountered, a list of the submodules for the module is displayed and the user specifies which are to be recompiled. Another set of keys is used to store this information.

After all information for the COMPILE mode has been entered, the user is asked if the RUN mode is desired. If RUN mode is desired, instructions starting with statement 260 are executed. If only the COMPILE mode is requested, a transfer is made to statement 365. Statements 365 through 430 write the selected files to file THEJOB based on the previously stored keys. In this case, all files would be from the NEWCANOF catalog. A transfer is then made to statement 660 where termination of the job is accomplished.

Statement 260

A call is made to the FILSEL entry in subroutine IDENT to retrieve the appropriate data base file and its characteristics. If the mode is not

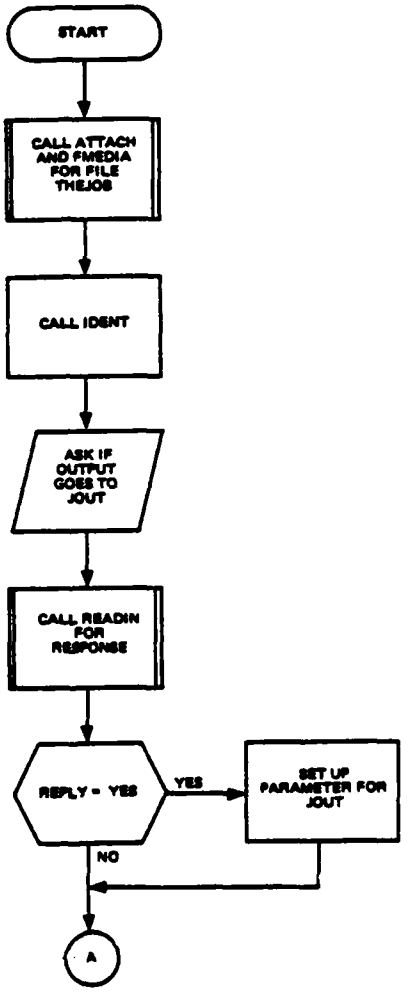


Figure 197. PROGRAM PERFORM (Part 1 of 18)

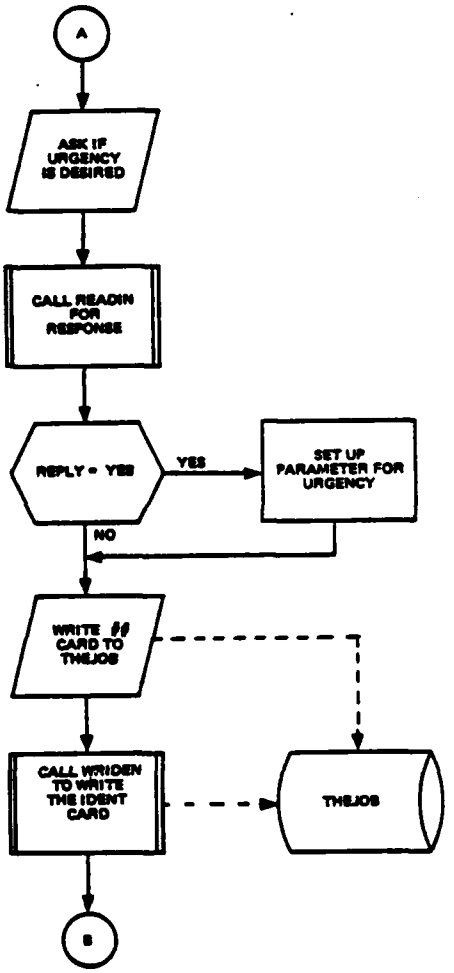


Figure 197. (Part 2 of 18)

THIS PAGE INTENTIONALLY LEFT BLANK

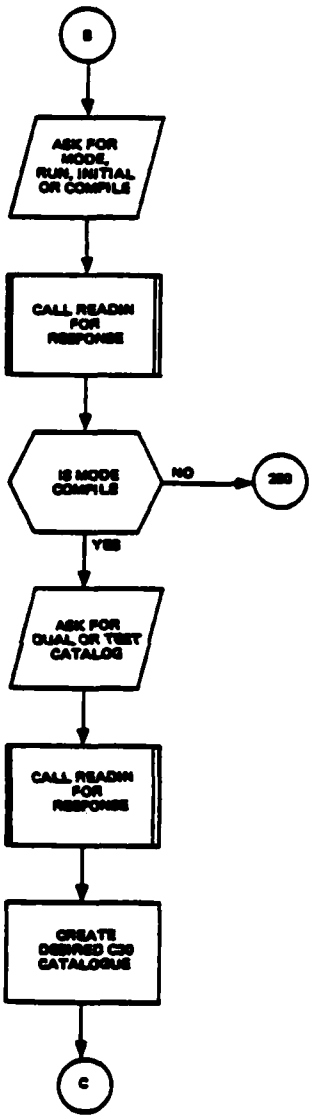


Figure 197. (Part 3 of 18)

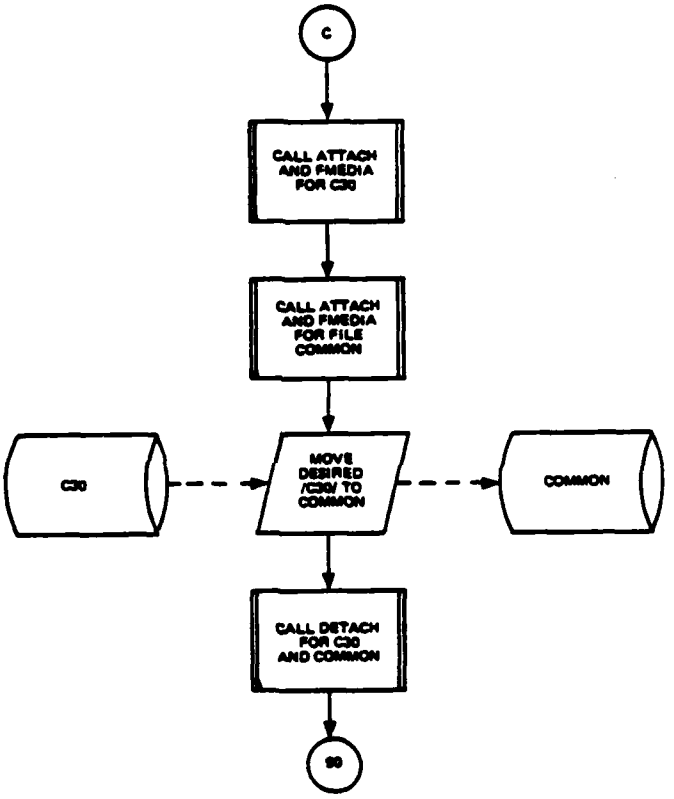


Figure 197. (Part 4 of 18)

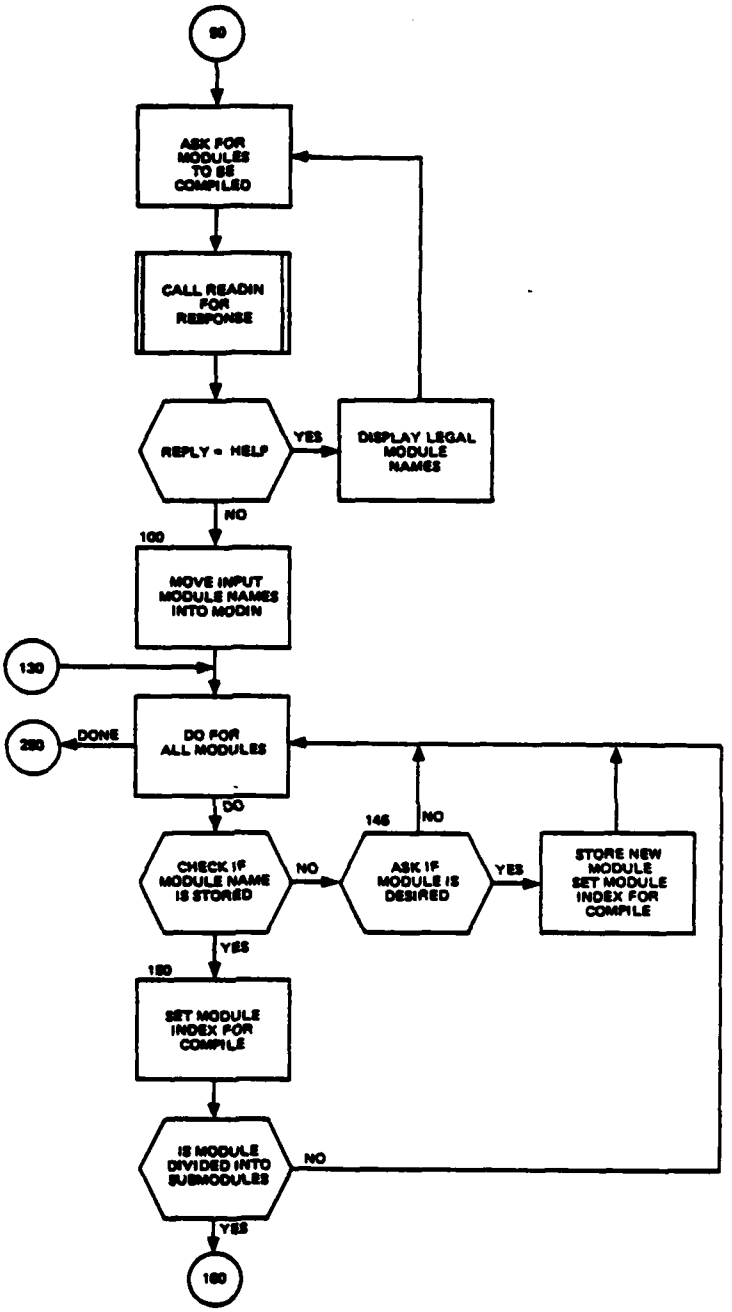


Figure 197. (Part 5 of 18)

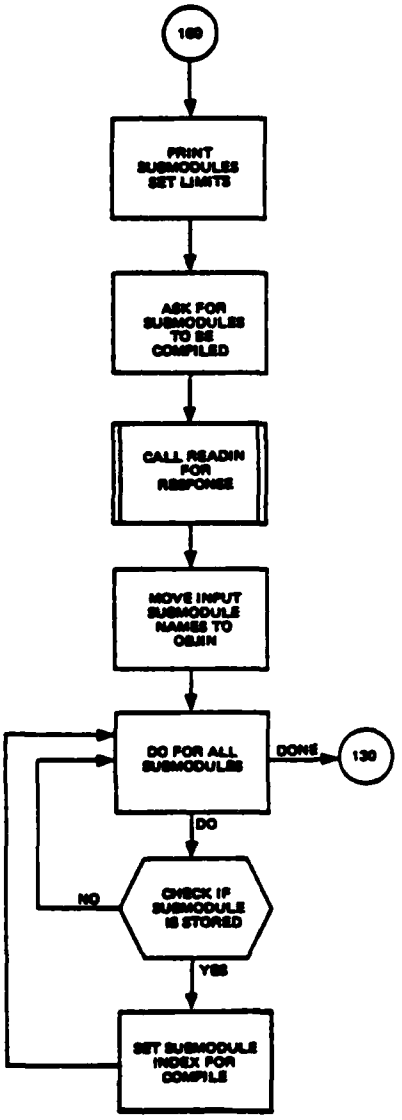


Figure 197. (Part 6 of 18)

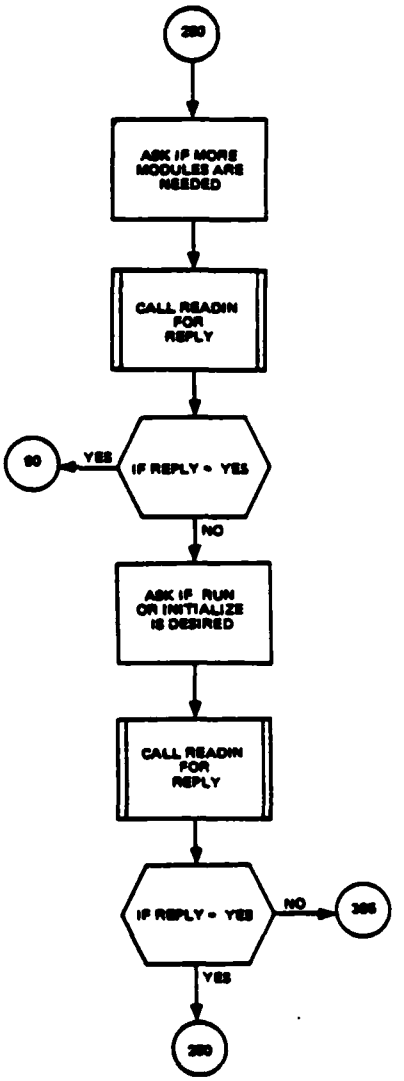


Figure 197. (Part 7 of 18)

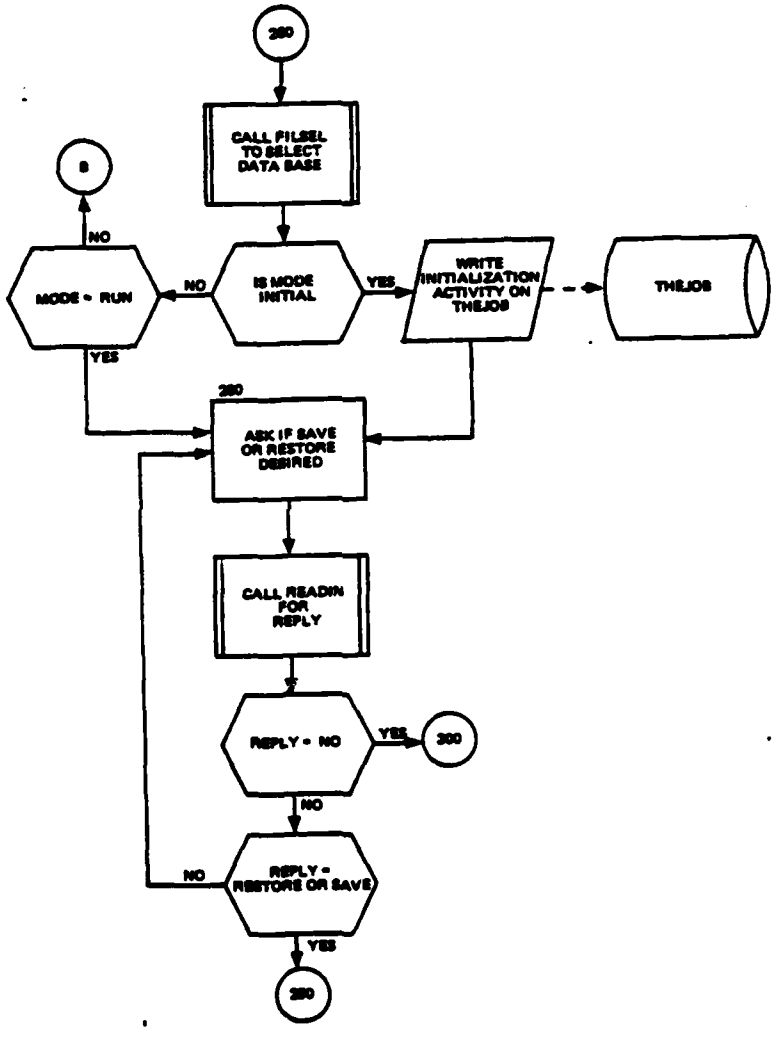


Figure 197. (Part 8 of 18)

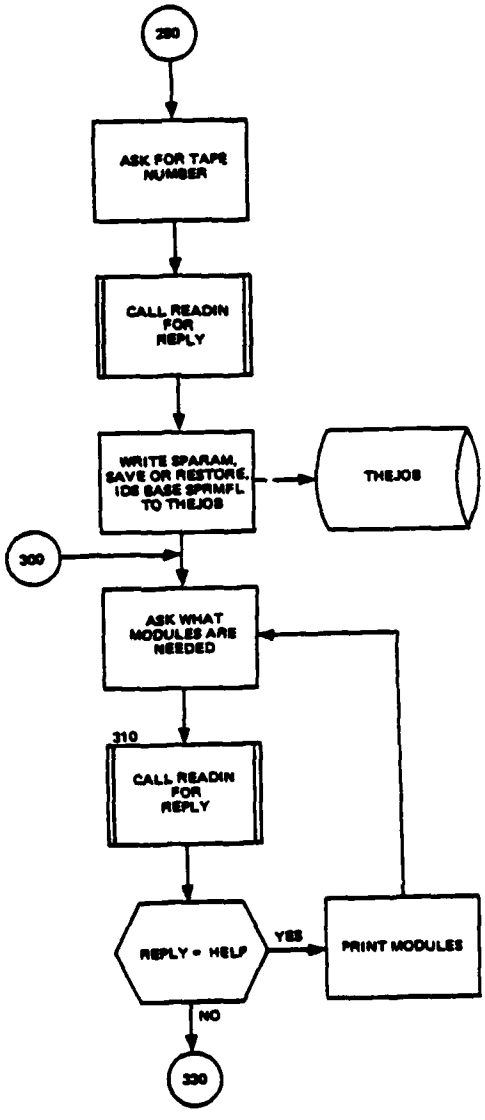


Figure 197. (Part 9 of 18)

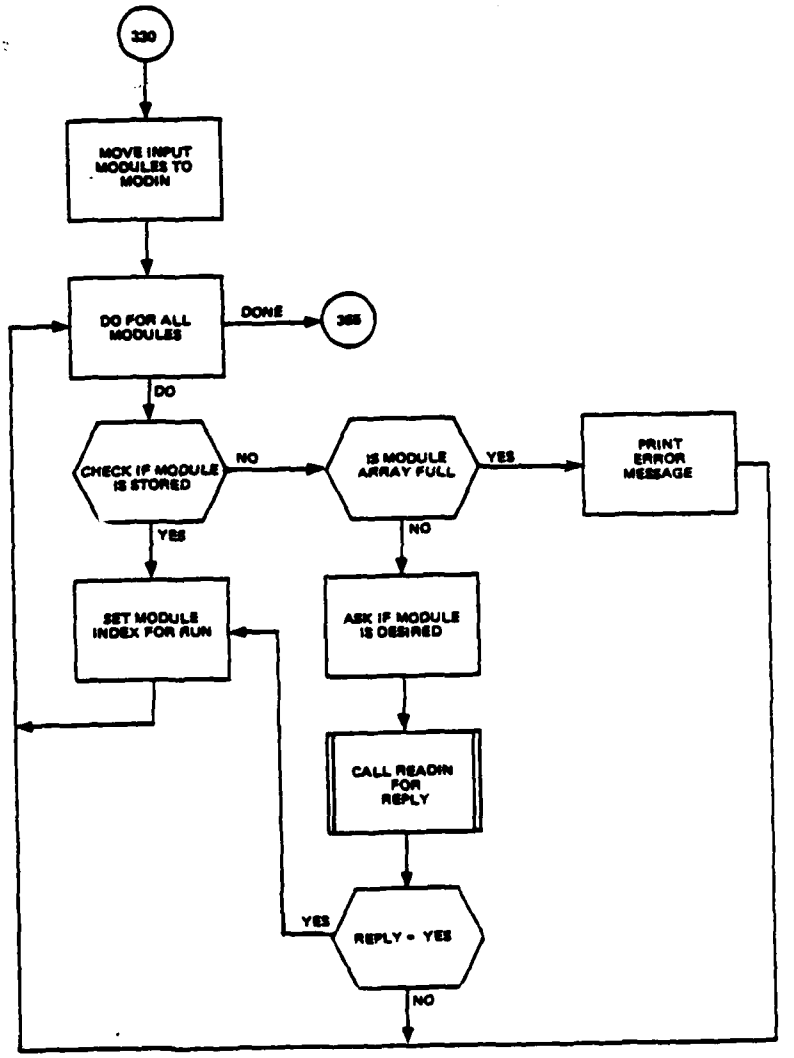


Figure 197. (Part 10 of 18)

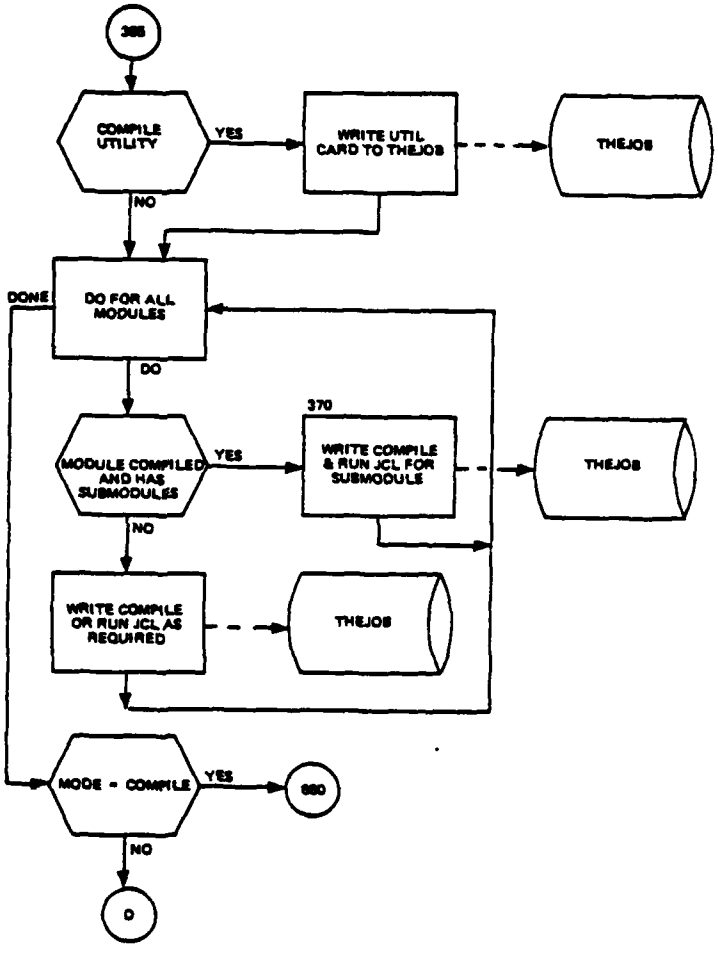


Figure 197. (Part 11 of 18)

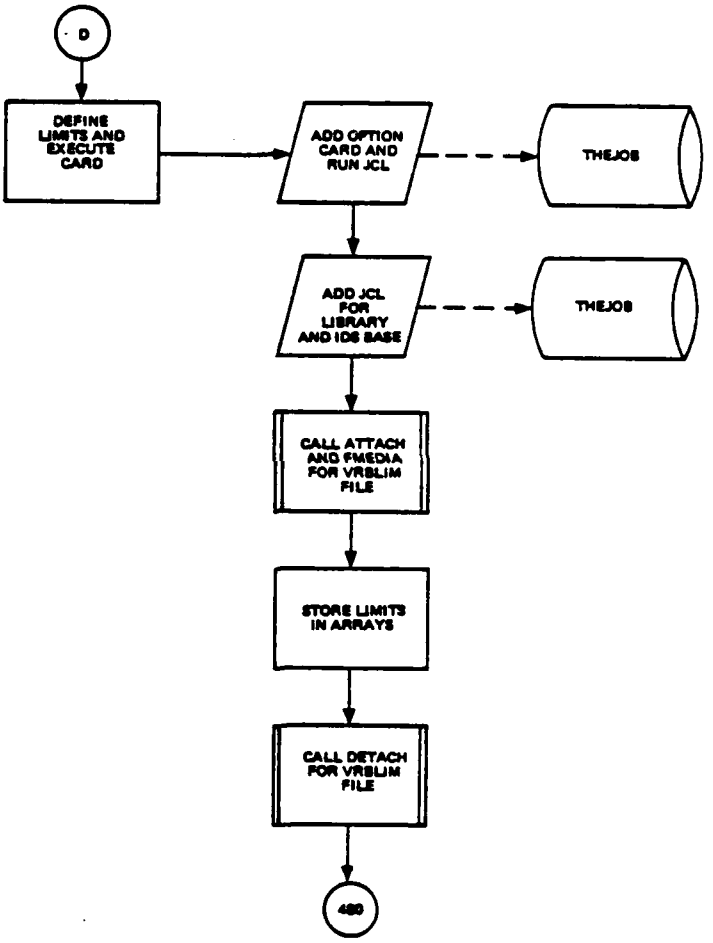


Figure 197. (Part 12 of 18)

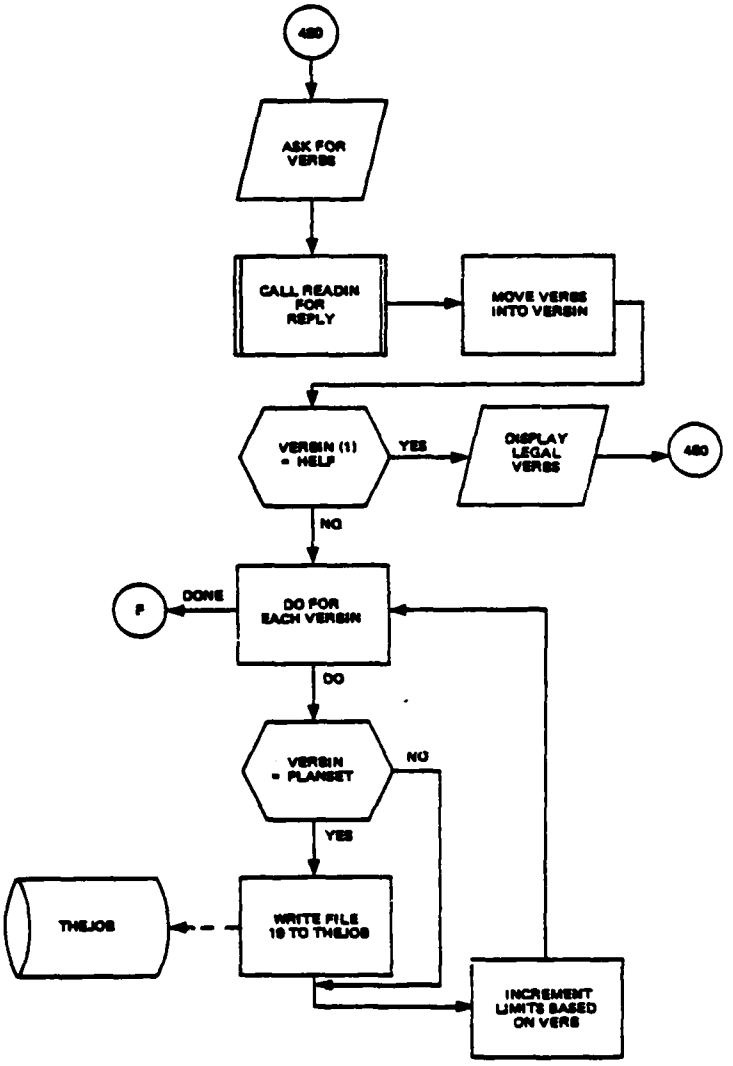


Figure 197. (Part 13 of 18)

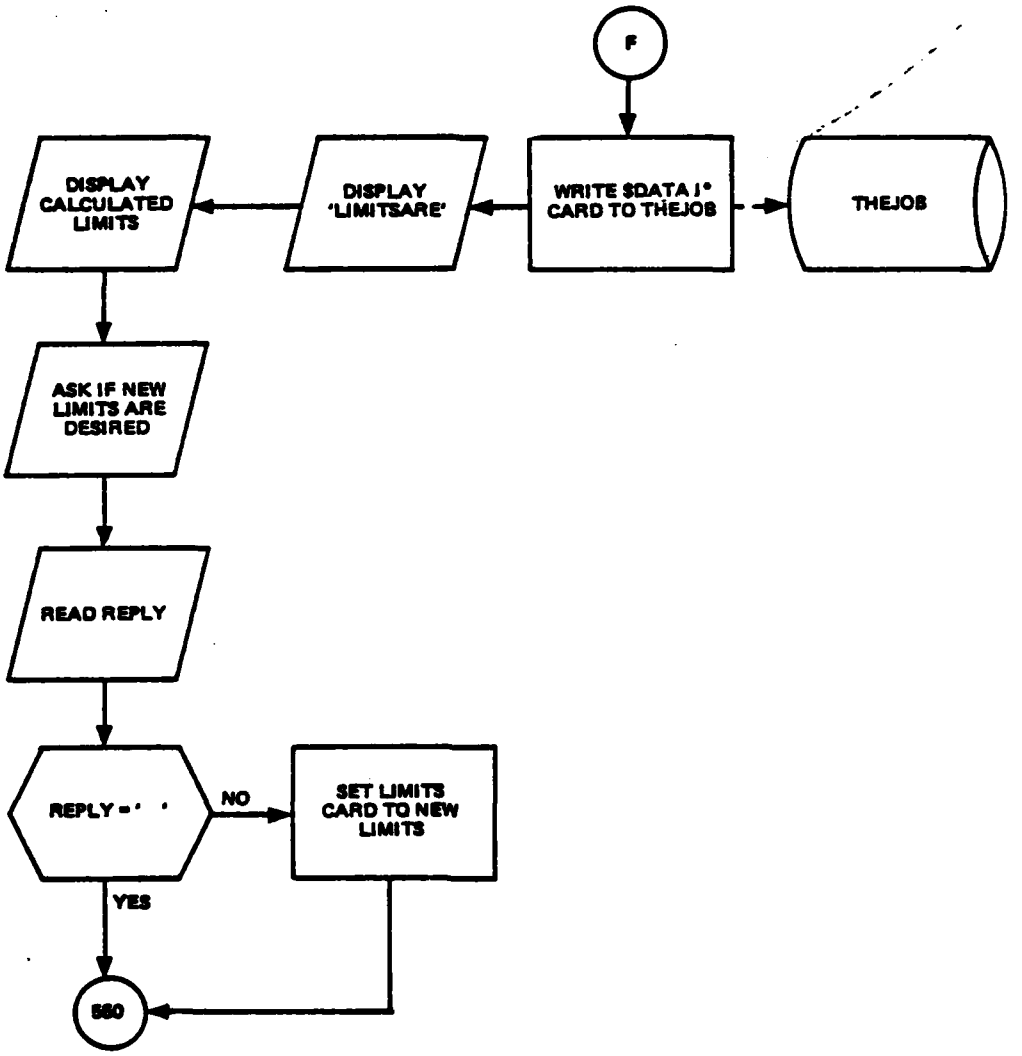


Figure 197. (Part 14 of 18)

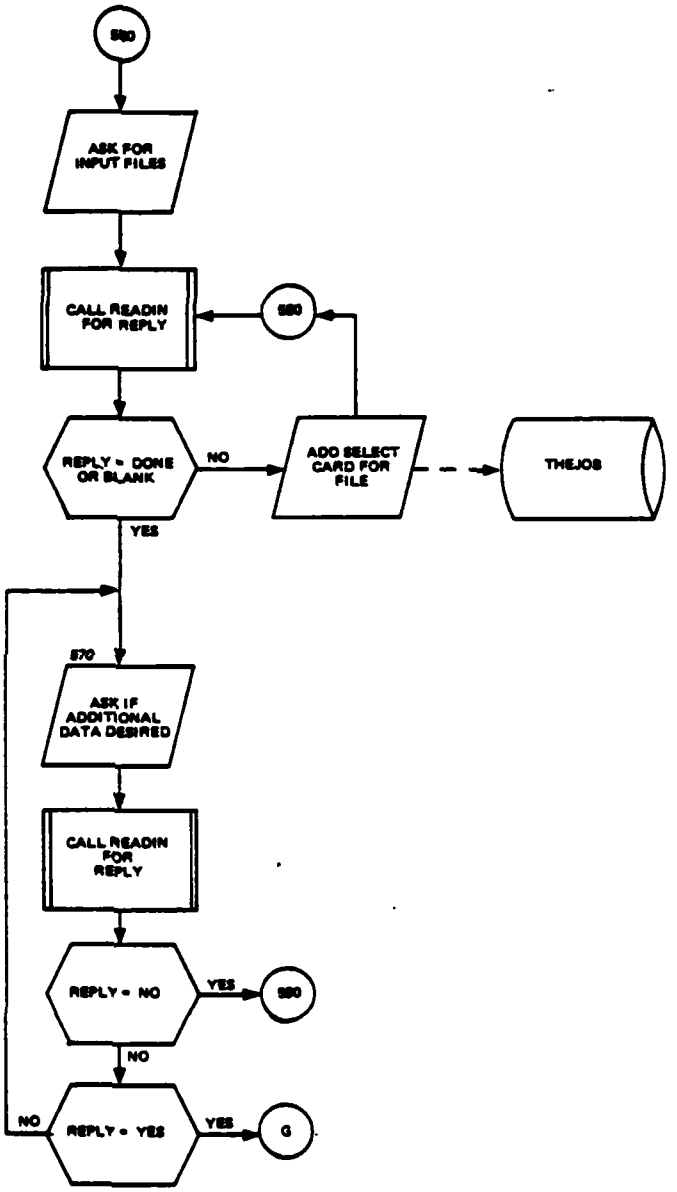


Figure 197. (Part 15 of 18)

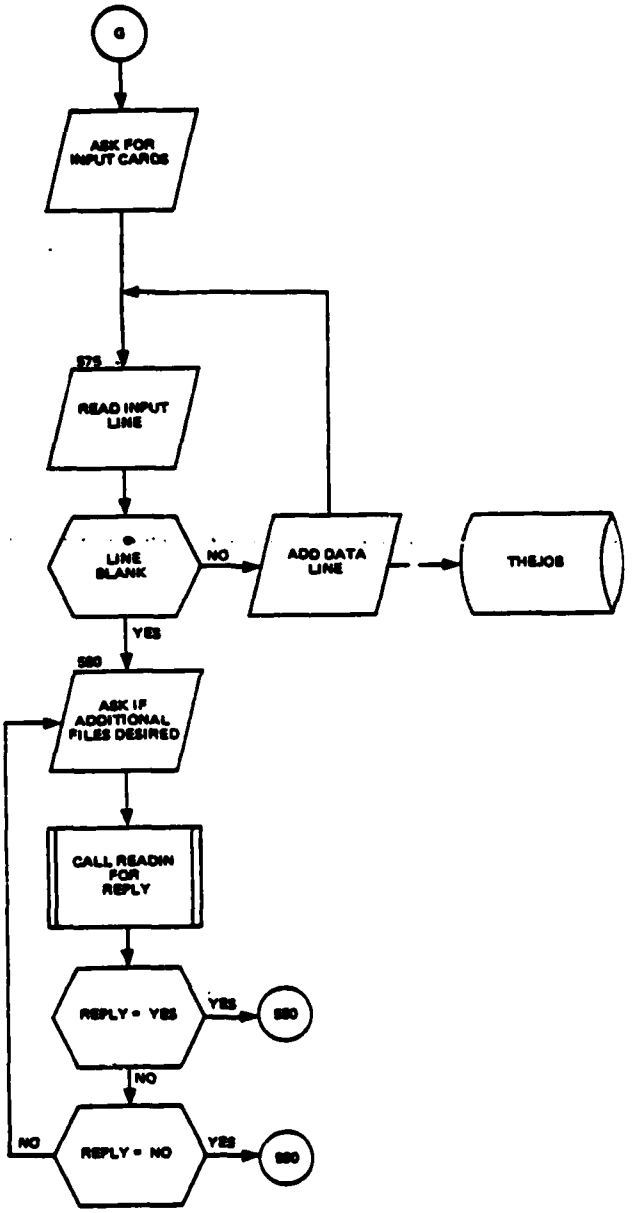


Figure 197. (Part 16 of 18)

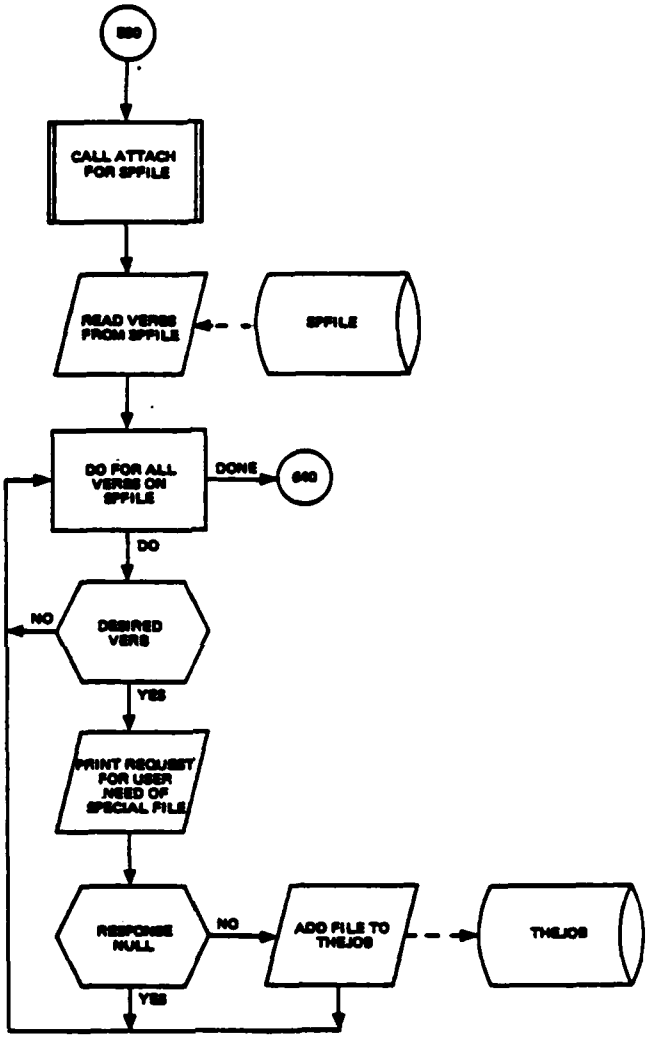


Figure 197. (Part 17 of 18)

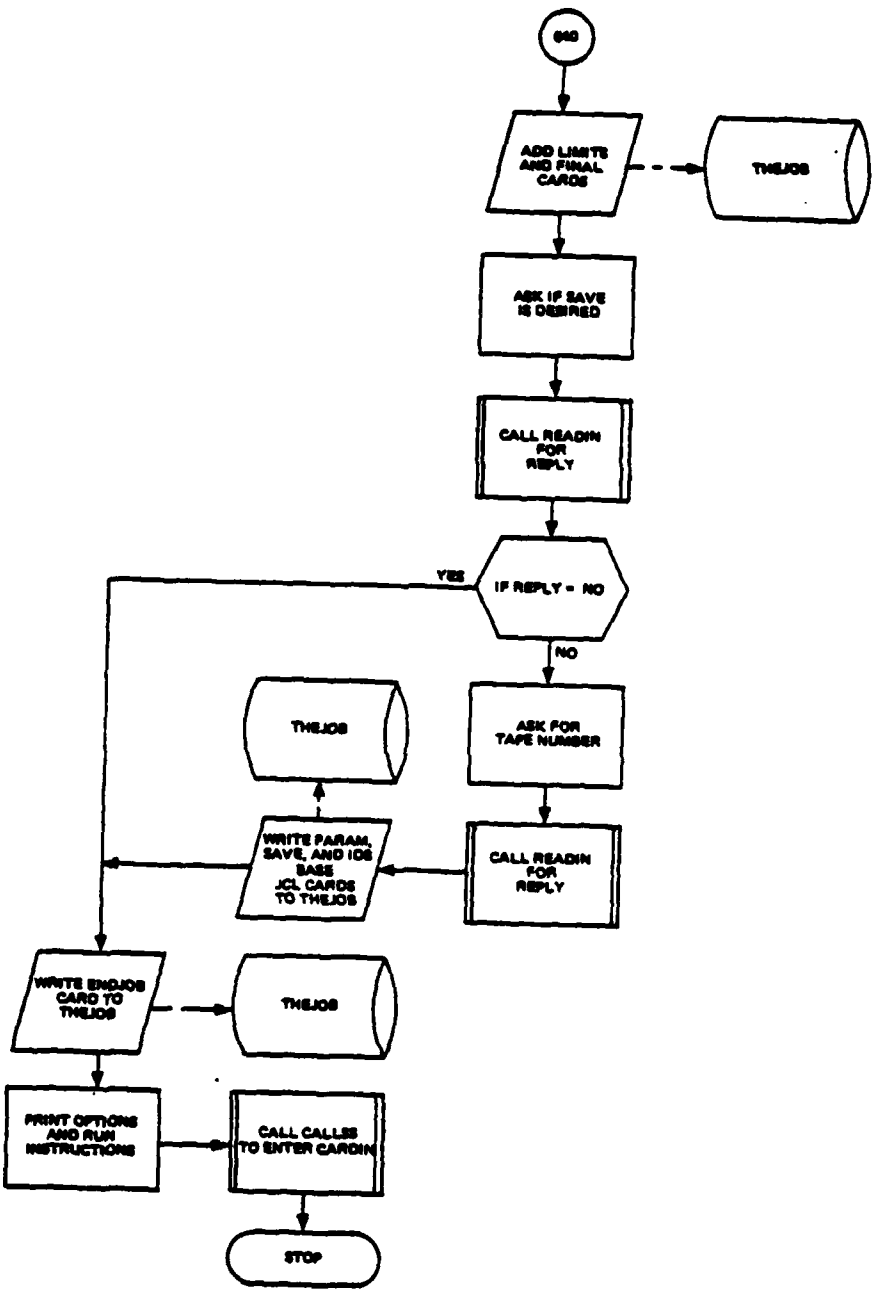


Figure 197. (Part 18 of 18)

INITIAL, a transfer is made to statement 270. The INITIAL mode writes to THEJOB file the JCL required to initialize the selected I-D-S data base. A transfer is then made to statement 280.

Statement 270

Statement 270 clears the screen prior to executing statement 280.

Statement 280

The user can select a preexecution RESTORE or SAVE of the I-D-S data base. If neither are desired, a transfer is made to statement 300. A parameter card containing the required tape number and the appropriate $ SELECT card are written to THEJOB file if SAVE or RESTORE are selected. Statement 300 is then executed.

Statement 300

If the COMPILE and RUN options have been selected, the user is asked to input any other modules needed in addition to those entered in the COMPILE mode. If only the RUN mode was selected, the user is asked to input the required modules. A response of HELP will result in a listing of the stored modules which can be selected.

An appropriate key is set to indicate the desired modules. A key is also set for the three modules required on every QUICK run. If a new module is encountered, the user will be asked to verify if the module is desired, the key will be set, and the module name will be stored. Only three new names can be entered from the COMPILE section and the RUN section during any single run. The required JCL for both the COMPILE and the RUN modes is then written to THEJOB starting at statement 360.

Statement 360

The keys for each module (stored and added) are checked and based on the value of the key, a CANOF or NEWCANOF catalog file string is written to THEJOB file. If the ALOC or PLANOUT modules were selected during the COMPILE phase, NEWCANOF catalogs are used for the submodules selected for compilation and CANOF modules are used for all of the other submodules in the requested module. At statement 430, a transfer is made to statement 660 if the COMPILE mode has been selected.

Statement 430

The standard FORTRAN execution JCL and the basic file definition for the QUICK system are then written to THEJOB file. A JCL card for the data base is output to THEJOB file. Statement 440 is then executed.

Statement 440

The VRBLIM file is attached, read in, and detached. The user then inputs a list of the desired verbs. If the reply is HELP, the legal verbs are listed. The program then calculates the run limits. The limits are displayed and the user is given the opportunity to change them. Next, the user is asked to input data file names. Each file name is added to THEJOB. When the reply DONE is encountered, the user is asked if additional data are desired. If so, the user is asked to input card images which are added to THEJOB. When a blank card is input, the user is asked if additional files are desired, and if so, control returns to the input of data files.

When no more input is desired, the program reads file SPFILE for any special files that the input verbs may require. The user is asked to select the JCL format for any such special file. The terminal part of the program (statement 640) is then executed.

Statement 640

The LIMITS card and the standard QUICK temporary files are written to THEJOB file. The user is then asked if a postexecution save of the I-D-S data base is desired. If a save is required, a parameter card containing the reel number of the save tape and a SAVE card is written to THEJOB file. Statement 660 is then executed.

Statement 660

The $ ENDJOB card is placed on THEJOB file. The key parameters of the completed job stream JCL are displayed for the user with instructions on how to execute the job. A call is then made to the CALLSS system and CARDIN is entered to facilitate running the job.

C.7 Subroutine READIN

PURPOSE: To convert letters from lower case to upper case

ENTRY POINTS: READIN

FORMAL PARAMETERS:

V:	Variable in which first eight characters may be returned
FLAG:	=1, return first eight characters in V ≠1, do not alter V

COMMON BLOCKS: LINE

SUBROUTINES CALLED: NONE

CALLED BY: PERFORM

Method:

LINE is filled from the user input replay. Each character is then examined and converted to uppercase if it is lowercase. FLAG is checked and if equal to 1, the first eight positions of LINE are encoded into V.

Subroutine READIN is illustrated in figure 198.

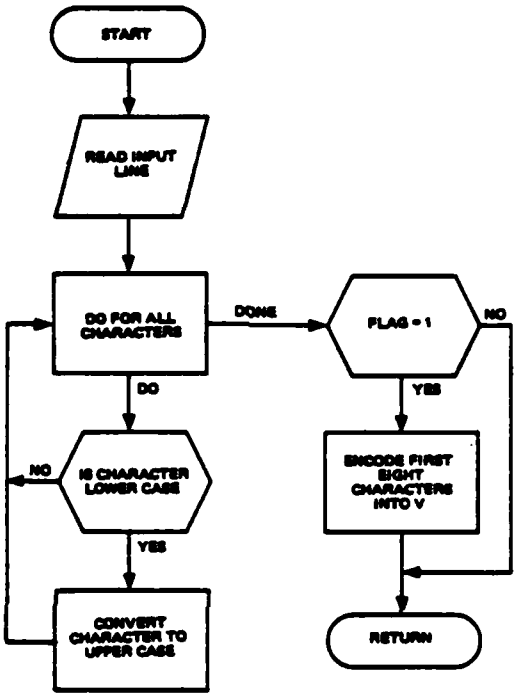


Figure 198. Subroutine READIN

C.8 Subroutine IDENT

PURPOSE: To build the JCL file IDENT card and maintain the IDENT2 file. Entry point WRIDEN writes the IDENT card and Entry FILSEL selects the data base.

ENTRY POINTS: IDENT, WRIDEN, FILSEL

FORMAL PARAMETERS:

WHO:	Name of user used in terminal output
FILE:	Catalog file string of I-D-S data file
CAT:	Subcatalog for program source
SIZE:	Size of I-D-S file in pages

COMMON BLOCKS: NONE

SUBROUTINES CALLED: USRCOD

CALLED BY: PERFORM

Method:

First the system subroutine USRCOD is called to obtain the USERID with which the user signed onto the system. This is compared with the list of USERIDs found on file UMC/PERFORM/IDENT2. If found, the information on IDENT2 is used to construct a $ IDENT card for the JCL file THEJOB. The other information needed for the parameters which are returned (i.e., WHO, FILE, CAT, and SIZE) is also obtained directly from the IDENT2 file. In the process, the user may be asked to select from several possible data files. Selection of these files is accomplished by the subroutine FILSEL which is called by PERFORM.

If the USERID does not appear on IDENT2, a series of questions and answers is used to build a new file entry before processing continues.

Subroutine IDENT is illustrated in figure 199.

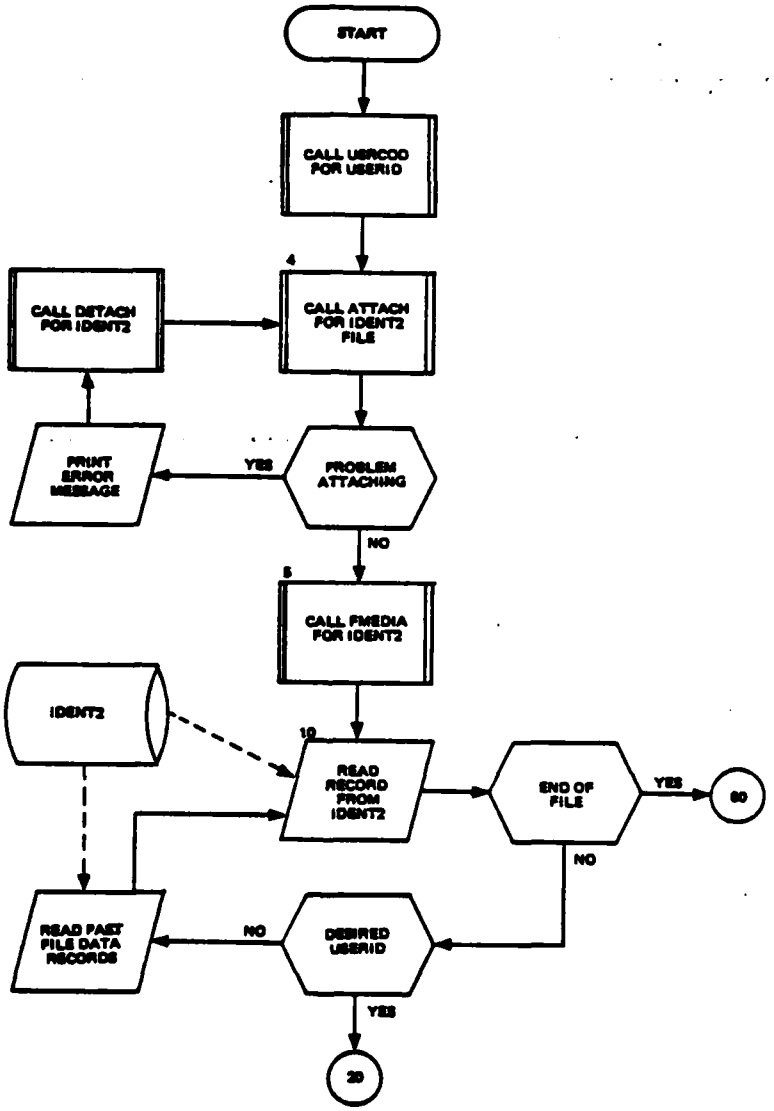


Figure 199. Subroutine IDENT (Part 1 of 5)

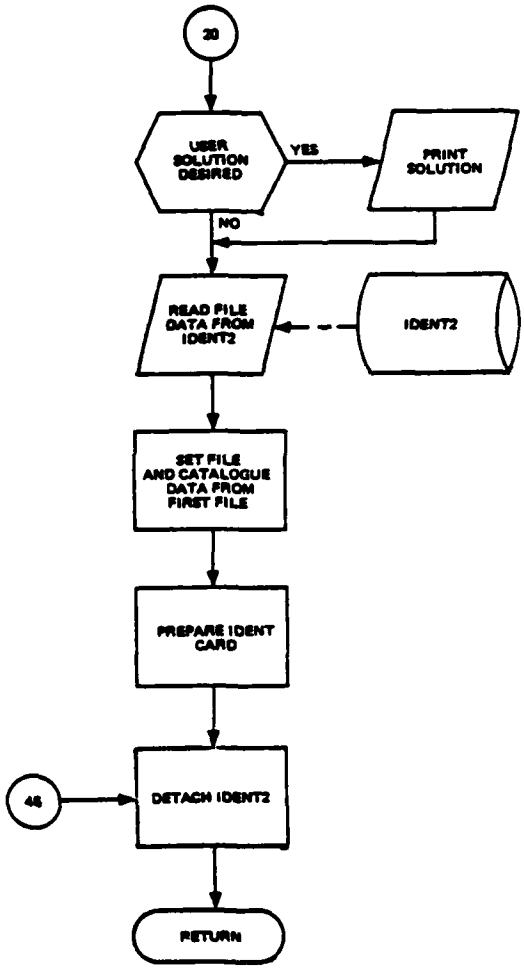


Figure 199. (Part 2 of 5)

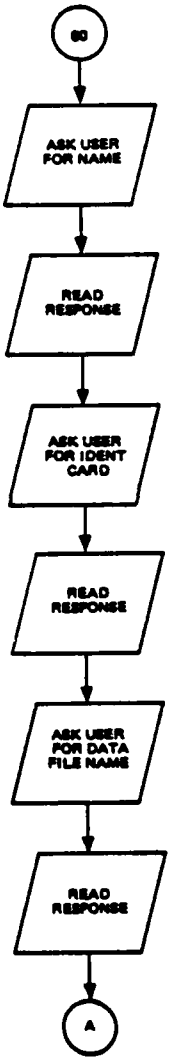


Figure 199. (Part 3 of 5)

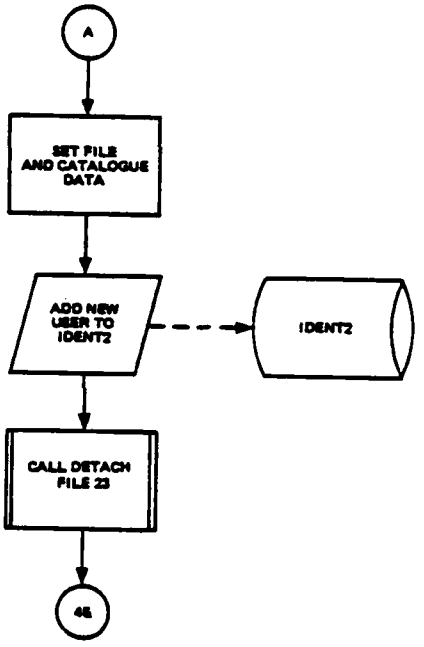


Figure 199. (Part 4 of 5)

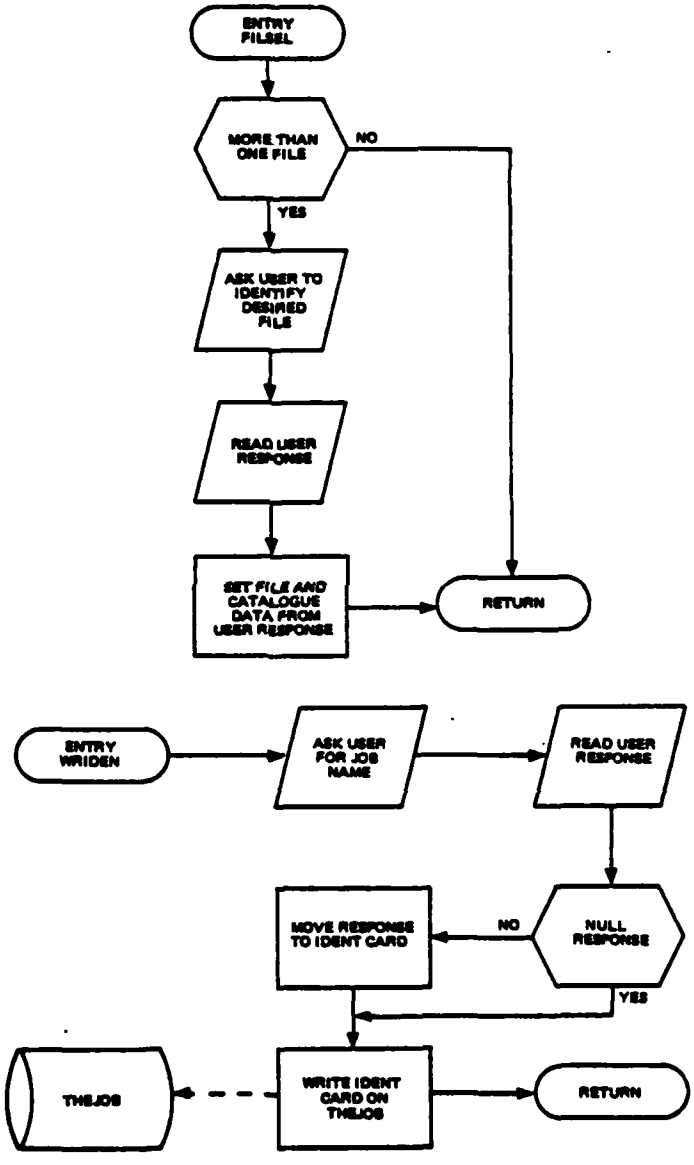


Figure 199. (Part 5 of 5)

THIS PAGE INTENTIONALLY LEFT BLANK

DISTRIBUTION

Addressee	Copies
CCTC Codes	
C124 (Reference and Record Set)	3
C124 (Stock)	6
C126	2
C313	1
C314	7
C630	1
DCA Code	
205	1
EXTERNAL	
Chief, Studies, Analysis and Gaming Agency, OJCS ATTN: SFD, Room 1D935, Pentagon, Washington, DC 20301	2
Chief of Naval Operations, ATTN: OP-654C, Room BE781 Pentagon, Washington, DC 20350	2
Commander-in-Chief, North American Air Defense Command ATTN: NPXYA, Ent Air Force Base, CO 80912	2
U.S. Air Force Weapons Laboratory (AFSC) ATTN: AFWL/SUL (Technical Library), Kirtland Air Force Base, NM 87117	1
Director, Strategic Target Planning, ATTN: (JPS), Offutt Air Force Base, NE 68113	2
Defense Technical Information Center, Cameron Station, Alexandria, VA 22314	12
	42

THIS PAGE INTENTIONALLY LEFT BLANK

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (when data entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CSM MM 9-77 Volume I, Parts I & II	2. GOVT. ACCESSION NO. AD-A085 813	3. RECEIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THE CCTC QUICK-REACTING GENERAL WAR GAMING SYSTEM (QUICK), Program Maintenance Manual, Data Management Subsystem		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR (s) Dale J. Sanders, Paul F. M. Maykrantz, Jim M. Herrin, Edward F. Bersson		8. CONTRACT OR GRANT NUMBER (s) DCA 100-75-C-0019
9. PERFORMING ORGANIZATION NAME & ADDRESS System Sciences, Incorporated 4720 Montgomery Lane Bethesda, Maryland 20014		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME & ADDRESS Command and Control Technical Center Room BE-685, The Pentagon, Washington, DC 20301		12. REPORT DATE 1 June 1977 13. NUMBER OF PAGES 956
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASS/DOWNGRADING SCHEDULE
16 DISTRIBUTION STATEMENT (of this report)		
17 DISTRIBUTION STATEMENT (of the abstract entered in block 20, if different from report) Approved for public release; distribution unlimited.		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (continue on reverse side if necessary and identify by block number) War Gaming, Resource Allocation		
20. ABSTRACT (continue on reverse side if necessary and identify by block number) The computerized Quick-Reacting General War Gaming System (QUICK) will accept input data, automatically generate global strategic nuclear war plans, provide statistical output summaries, and produce input tapes to simulator subsystems external to QUICK.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE 925 UNCLASSIFIED CH-3

SECURITY CLASSIFICATION OF THIS PAGE (when data entered)

20. ABSTRACT (Continued)

The Program Maintenance Manual consists of four volumes which facilitate maintenance of the war gaming system. This volume, Volume I, provides the programmer/analyst with a technical description of the purpose, functions, general procedures, and programming techniques applicable to the modules and subroutines of the Data Management Subsystem.

The Program Maintenance Manual complements the other QUICK Computer Manuals to facilitate application of the war gaming system. These manuals (Series 9-77) are published by the Command and Control Technical Center (CCTC), Defense Communications Agency (DCA), The Pentagon, Washington, DC 20301.


FIL